



ugr

Universidad
de Granada

TRABAJO FIN DE MASTER
MÁSTER UNIVERSITARIO EN DESARROLLO DE SOFTWARE

Internet de las Cosas basado en tecnologías de Google

Autor

Victor Cruz Gomez

Directores

Prof. Dr. Juan Antonio Holgado Terriza



Escuela Técnica Superior de Ingenierías Informática y de
Telecomunicación

—
Granada, septiembre de 2017

Dedicatoria

A mi esposa María del Carmen por todo el apoyo brindado desde el día que nos conocimos y que aún lo sigue haciendo, y a mi bella hija Grace Adriana quien me da fuerzas para seguir adelante.

A mis queridos padres, German y Miriam, a mi padre por sus consejos de amigo que siempre me alentaron a seguir adelante y no mirar atrás, a mi señora madre por su apoyo incondicional quien me enseño desde niño a seguir adelante a pesar de las adversidades.

A mis hermanos Julia, Richard y mi sobrino Brandon por su apoyo y confianza en las tareas que me propuse realizar.

A mi tío Jhonny (†) a quien le hubiera gustado verme en este momento, ya que fue una persona muy importante en mi vida.

A toda mi familia gracias por alentarme y apoyarme para seguir adelante.

Gracias por todo...

Agradecimientos

A Dios por darme la vida y salud.

A mi familia que se constituye en un pilar importante en mi vida.

Al Prof. Dr. Juan Antonio Holgado quien como tutor me oriento en el desarrollo del presente proyecto.

Al Ministerio de Educación de Bolivia por darme la oportunidad de formar parte del proyecto denominado “100 Becas de estudio para la soberanía científica y tecnológica” y gracias a ello realizar este Master en Desarrollo de Software de la Universidad de Granada, España.

Gracias a todos...

Resumen

Los avances de la tecnología para el Internet de las Cosas es toda una realidad con muchas promesas por delante gracias a las tecnologías que se han ido desarrollando en los últimos años. Desde la acuñación de este concepto y debido a su enorme potencial, el ámbito tecnológico puso su interés en el Internet de las Cosas. Por ello últimamente importantes compañías internacionales han exhibido diferentes propuestas tecnológicas. Un de ellas ha sido Google, que ha dado a conocer un ecosistema de tecnologías para la implementación de sistemas del Internet de las Cosas.

Dentro del contexto de Internet de las Cosas, en el presente trabajo fin de master, se recopiló información relacionado al mismo, para tener un panorama actual de este concepto, tales como: definiciones, antecedentes históricos, arquitecturas, elementos y las tecnologías utilizadas en la implementación de un producto del Internet de las Cosas. Asimismo, se estudiaron las tecnologías de Google propuestas para la creación de productos del Internet de las Cosas, específicamente: Android Things que es un sistema operativo; Weave una plataforma de comunicación para dispositivos; y Google Cloud Platform una plataforma de computación en la nube.

Como caso de estudio, para la implementación de un producto del Internet de las Cosas, se desarrolló un sistema medioambiental basados en tecnologías de Google, como apoyo al estudio del efecto medioambiental sobre la salud de las personas.

El resultado obtenido fue un sistema medioambiental diseñado con tecnologías de Google. La arquitectura del sistema fue implementada en tres capas: una capa de detección, una de computación en la nube y otra de aplicación. La capa de detección fue desarrollada con Sistemas Empotrados en Raspberry PI gestionado por Android Things; la capa de computación en la nube fue implementada en Google Cloud Platform; y finalmente en la capa de aplicación se desarrolló un programa móvil Android para la interacción con el usuario.

Abstract

The advances in technology for the Internet of Things is a reality with many promises ahead thanks to the technologies that have been developed in recent years. Since the coinage of this concept and due to its enormous potential, the technological field put its interest in the Internet of Things. That is why lately important international companies have exhibited different technological proposals. One of them has been Google, which has unveiled an ecosystem of technologies for the implementation of Internet of Things systems.

Within the context of the Internet of Things, in the present master's degree project, information related to it was compiled, to have a current view of this concept, such as: definitions, historical background, architectures, elements and the technologies used in the implementation of an Internet of Things product. Likewise, the proposed Google technologies for the creation of Internet of Things products were studied, specifically: Android Things, which is an operating system; Weave a communication platform for devices; and Google Cloud Platform, a cloud computing platform.

As a case study, for the implementation of an Internet of Things product, an environmental system based on Google technologies was developed to support the study of the environmental effect on people's health.

The result obtained was an environmental system designed with Google technologies. The architecture of the system was implemented in three layers: a detection layer, a cloud computing layer and an application layer. The detection layer was developed with Embedded Systems in Raspberry PI managed by Android Things; the cloud computing layer was implemented in Google Cloud Platform; and finally, in the application layer, an Android mobile program was developed for interaction with the user.

ÍNDICE DE CONTENIDOS

Capítulo 1	1
Introducción.....	1
1.1 Motivación	1
1.2 Objetivos.....	5
1.2.1 Objetivo general.....	5
1.2.2 Objetivos específicos.....	5
1.3 Estructura del Trabajo.....	6
Capítulo 2	8
Internet de las Cosas	8
2.1 Antecedentes históricos	8
2.2 Internet de las Cosas.....	10
2.3 Definiciones	11
2.4 Elementos del Internet de las Cosas.....	12
2.4.1 Identificación	13
2.4.2 Detección.....	13
2.4.3 Comunicación	14
2.4.4 Computación	14
2.4.5 Servicios	15
2.4.6 Semántica	16
2.5 Arquitectura.....	16
2.5.1 Arquitectura de 3 capas.....	17
2.5.2 Arquitectura basada en middleware	18
2.5.3 Arquitectura basada en SOA	19
2.5.4 Arquitectura de 5 capas.....	21
2.6 Tecnologías.....	22
2.6.1 Identificación	23
2.6.2 Detección.....	24
2.6.3 Comunicación	25
2.6.4 Computación	29
Capítulo 3	34

Internet de las Cosas y tecnologías de Google	34
3.1 Android Things	34
3.2 Weave	34
3.3 Google Cloud Platform	34
3.4 Arquitectura.....	35
3.4.1 IoT basado en Weave.....	35
3.4.2 IoT basado Google Cloud Platform.....	36
3.5 Fabricantes y clientes asociados a Google.....	37
Capítulo 4.....	39
Android Things.....	39
4.1 ¿Que es Android Things?	39
4.2 Evolución	39
4.3 Requisitos de Hardware y Software.....	40
4.3.1 Hardware.....	40
4.3.2 Software	42
4.4 Arquitectura de Android Things.....	45
4.5 Características	47
4.5.1 Paquetes de aplicaciones.....	47
4.5.2 Compatibilidad con los servicios de Google	48
4.5.3 Seguridad	50
4.5.4 Pantallas opcionales	50
4.5.5 Notificaciones.....	51
4.5.6 Controladores de Usuarios	51
4.5.7 Periféricos de Entrada y Salida.....	52
4.6 Implementación de una aplicación para Android Things.....	59
Capítulo 5.....	65
IoT basado en Weave	65
5.1 ¿Qué es Weave?.....	65
5.2 Requisitos de Hardware y Software.....	65
5.2.1 Hardware.....	65
5.2.2 Software	66

5.3	Seguridad de Weave.....	68
5.4	Dispositivo Weave	68
5.5	Funcionalidad y componentes de dispositivo	69
5.6	Información Obligatoria	69
5.7	Esquema común de dispositivos.....	70
5.8	Guía de Dispositivos	71
5.9	Desarrollo de dispositivo	73
	Capítulo 6.....	75
	IoT basado en Google Cloud Platform.....	75
6.1	¿Qué es Google Cloud Platform?	75
6.2	Requisitos de Hardware y Software.....	76
6.3	Componentes de Google Cloud Platform.....	76
6.3.1	Procesamiento.....	76
6.3.2	Almacenamiento y base de datos	77
6.3.3	Redes	78
6.3.4	Big Data.....	79
6.3.5	Internet de las Cosas.....	80
6.3.6	Aprendizaje Automático.....	81
6.3.7	Identidad y Seguridad	83
6.4	Implementación de servicios en Google Cloud Platform	83
6.4.2	Publicación y Suscripción de mensajes en Cloud Pub/Sub	84
6.4.3	Contar las palabras de los libros de Shakespeare en Cloud Dataflow	85
6.4.4	Consulta del historial revisiones de Wikipedia en BigQuery	88
	Capítulo 7.....	90
	Desarrollo de un sistema medioambiental basado en IoT.....	90
7.1	Descripción del problema	90
7.2	Caso de estudio TEMPRE: Sistema de apoyo para el estudio del efecto medioambiental sobre salud de las personas.	93
7.3	Arquitectura.....	94
7.3.1	Capa de Detección	94
7.3.2	Capa de Computación en la nube	95

7.3.3	Capa de Aplicación	95
7.4	Requisitos de Hardware y Software.....	96
7.4.1.1	Hardware.....	96
7.4.1.2	Software	96
7.5	Desarrollo de la Capa de Detección.....	97
7.5.1	Ensamblado del Hardware.....	98
7.5.2	Instalación de Android Things en Raspberry Pi.....	99
7.5.3	Conexión entre la computadora y el Sistema Empotrado.	100
7.5.4	Creación de la aplicación Android Things	101
7.5.5	Despliegue de la aplicación Android Things.....	110
7.6	Desarrollo de la Capa de Computación en la nube.....	110
7.6.1	Implementación de Google Cloud Platform	111
7.6.2	Activación de la cuenta de servicio	111
7.6.3	Creación del proyecto Google Cloud Platform	112
7.6.4	Servicio Cloud Pub/Sub.....	113
7.6.5	Servicio Cloud Dataflow.....	116
7.6.6	Servicio BigQuery	122
7.6.7	Servicio Compute Engine.....	125
7.6.8	Configuración del servicio Firebase Cloud Messaging.....	128
7.7	Desarrollo de la Capa de Aplicación.....	130
7.7.1	Implementación de la aplicación del usuario	131
7.8	Fotografías del Sistema	135
Capítulo 8.....		139
Conclusiones		139
8.1	Conclusiones.....	139
8.2	Problemas presentados.....	145
8.3	Análisis de los objetivos	147
8.4	Trabajos futuros	148
Referencias Bibliográficas.....		150
Anexos.....		154
Anexo A - Guías para dispositivos Weave.....		154

Anexo B - Google Cloud Platform	157
Anexo C - Herramientas y tecnologías utilizadas en el proyecto	160
Anexo D - Sistemas Operativos IoT de Código Abierto.....	163

ÍNDICE DE FIGURAS

Figura 1: Visión del IoT [10].....	11
Figura 2: Elementos del IoT [20].....	13
Figura 3: Propuestas de arquitecturas para el IoT [20].....	17
Figura 4: Integración Android Things - Weave - Google Cloud Platform [9].....	36
Figura 5: Objetos funcionando con Android Things [52].....	39
Figura 6: Arquitectura Android Things [54].....	45
Figura 7: Capas gestionados por Google y desarrolladores [54].....	51
Figura 8: Pines de entrada y salida de propósito general [56].....	53
Figura 9: Diagrama de serie de pulso entre el 0%, 25% y 100% [54].....	54
Figura 10: Conexión I2C [54].	56
Figura 11: Conexión SPI [54].	57
Figura 12: Conexión UART [54].....	58
Figura 13: Diseño del prototipo [56]	60
Figura 14: Montaje del circuito.....	64
Figura 15: Objetos conectados a través de Weave [57]	65
Figura 16: Consola de Desarrollador IoT [50]	67
Figura 17: Arquitectura del servicio Pub/Sub.....	84
Figura 18: Proceso de extracción, transformación y carga.....	86
Figura 19: Argumentos de conexión: Cuenta, ID Proyecto y ubicación del Cloud Storage.	
.....	87
Figura 20: Vista de la ejecución de los procesos de transformación de datos [59].	88
Figura 21: Menú de acceso al servicio de BigQuery [59].....	88

Figura 22: Resultado de la ejecución de una consulta BigQuery [59].....	89
Figura 23: Arquitectura global del Sistema Medioambiental.....	93
Figura 24: Arquitectura en capas del Sistema Medioambiental.....	94
Figura 25: Capa de Detección.....	97
Figura 26: Ensamblado del Hardware.....	98
Figura 27: Visualización de la dirección IP del Raspberry PI.....	101
Figura 28: Comando de conexión adb.....	101
Figura 29: Estructura de la aplicación Android Things.	102
Figura 30: Interfaz de la aplicación.	108
Figura 31: Capa de Computación en la nube.	111
Figura 32: Servicio Cloud Pub/Sub.....	113
Figura 33: Servicio Cloud Pub/Sub detallado.	114
Figura 34: Servicio Cloud Dataflow.....	116
Figura 35: Arquitectura interna del servicio Cloud Dataflow.	118
Figura 36: Estructura del proyecto Dataflow.	119
Figura 37: Objeto JSON publicado en el Cloud Pub/Sub.....	121
Figura 38: Servicio BigQuery.....	122
Figura 39: Arquitectura interna del Servicio BigQuery.....	123
Figura 40: Servicio Compute Engine.	125
Figura 41: Servicio Compute Engine detallado.....	125
Figura 42: Servicio Firebase Cloud Messaging.....	128
Figura 43: Servicio Firebase Cloud Messaging detallado.....	128
Figura 44: Capa de Aplicación.	131
Figura 45: Interfaz de registro de usuario.	132
Figura 46: Notificación de alerta.	133
Figura 47: Cuestionario facial de dolor.	134
Figura 48: Cuestionario para la detección de regiones con dolor.....	134
Figura 49: Sistema Empotrado en funcionamiento.	142
Figura 50: Capa de Computación en la nube.	143

ÍNDICE DE TABLAS

Tabla 1: Clasificación de tecnologías por elementos del IoT.....	23
Tabla 2: Fabricantes y clientes de Google Cloud Platform.....	38
Tabla 3: Plataforma de Hardware compatibles con Android Things [54].	40
Tabla 4: Especificación técnicas de las plataformas de Hardware.....	41
Tabla 5: Problemas conocidos en las versiones previas de Android Things.	42
Tabla 6: API soportado y no soportado por Android Things.	48
Tabla 7: Sensores para GPIO [56].	54
Tabla 8: Dispositivos con soporte PWM [56].....	55
Tabla 9: Protocolos de serie.	55
Tabla 10: Módulos con soporte I2C [56].....	56
Tabla 11: Módulos con soporte SPI [56].....	57
Tabla 12: Módulos con soporte UART [56].....	59
Tabla 13: Aplicación del desarrollador Weave [50].	68
Tabla 14: Esquema común para tomacorrientes [50].....	71
Tabla 15: Guía para termostatos [50].	72
Tabla 16: Creación de un tema y un suscriptor.....	85
Tabla 17: Pasos para la publicación de un mensaje desde consola.....	85
Tabla 18: Descarga de la imagen de Android Things.....	99
Tabla 19: Formateado del SD y descompresión de la imagen de Android Things.....	100
Tabla 20: Pasos para desplegar la aplicación.....	110
Tabla 21: Pasos para la activación de la cuenta de servicio.	112
Tabla 22: Pasos para la creación de un proyecto en Google Cloud Platform.....	113
Tabla 23: Pasos para habilitar la API Pub/Sub y creación de temas.	115
Tabla 24: Pasos para otorgar permisos de publicación.....	115
Tabla 25: Pasos para habilitar la API Cloud Dataflow.....	117
Tabla 26: Pasos para habilitar la API BigQuery.	124
Tabla 27: Pasos para otorgar permisos a la cuenta de Google.....	124
Tabla 28: Pasos para habilitar la API Compute Engine.	126

Tabla 29: Pasos para otorgar permisos a la cuenta de Google.....	127
Tabla 30: Pasos para crear la máquina virtual Debían 8.....	127
Tabla 31: Pasos para crear un proyecto en Firebase.....	129
Tabla 32: Pasos para obtener la clave del servidor	130
Tabla 33: Clasificación de tecnologías por elementos del IoT	140
Tabla 34: Interfaces de la aplicación de usuario.	145
Tabla 35: Guías para dispositivos Weave - bombillas.....	154
Tabla 36: Guías para dispositivos Weave - tomacorrientes.....	155
Tabla 37: Guías para dispositivos Weave - televisores.....	155
Tabla 38: Guías para dispositivos Weave - interruptores.....	156
Tabla 39: Servicios de Google Cloud Platform.	157
Tabla 40: Lista de Herramientas y tecnologías utilizados en el proyecto.....	160
Tabla 41: Tabla comparativa de Sistemas Operativos IoT de Código Abierto.	163

ÍNDICE DE CÓDIGOS

Código 1: AndroidManifest.xml.....	61
Código 2: Build.gradle.....	62
Código 3: BlinckActivity.java.....	62
Código 4: Método onCreate.	63
Código 5: Método onDestroy.....	63
Código 6: Objeto de la clase Runnable.	64
Código 7: Programa contador de palabras.	87
Código 8: AndroidManifest.xml.....	103
Código 9: IoTMedioambientalActivity.java - objeto SensorManager y Controladores ...	104
Código 10: IoTMedioambientalActivity.java - Obtención de la temperatura.....	104
Código 11: IoTMedioambientalActivity.java - Obtención de la presión.	104
Código 12: IoTMedioambientalActivity.java - Método onCreate.....	105

Código 13: PlacasHardware.java	106
Código 14: PublicadorPubsub.java.....	107
Código 15: main_activity.xml.....	107
Código 16: Credenciales.json.....	108
Código 17: Builde.gradle.....	109
Código 18: DataFlowSensorBigQuery.java.....	120
Código 19: DataFlowSensorBigQuery.java - Preferencias de tabla.	120
Código 20: DataFlowSensorBigQuery.java - Configuración de ejecución del programa.	121
Código 21: DataFlowSensorBigQuery.java - Extracción de datos.....	121
Código 22: DataFlowSensorBigQuery.java - Método principal del programa.	122

ÍNDICE DE GRÁFICOS

Gráfico 1: Grado de consecución de cada objetivo específico.....	147
--	-----

Capítulo 1

Introducción

1.1 Motivación

El presente trabajo viene motivado por el creciente protagonismo que la tecnología está adquiriendo en nuestras vidas. Vivimos en un mundo interconectado donde todos los días utilizamos nuestros dispositivos electrónicos (computadoras de escritorio, computadoras personales, teléfonos móviles, tabletas, etc.) para comunicarnos entre sí a través de Internet. Ya sea enviando un correo electrónico, un mensaje de texto, leer noticias, etc. Pero resulta que en los últimos años la comunicación ya no es solamente entre personas, sino que también las “cosas” están empezando a comunicarse con nosotros por medio de la dotación de sensores.

Por ejemplo, en el ámbito de la salud, se han fabricado pequeñas capsulas que al ser ingeridas permiten ayudar a los médicos a diagnosticar y determinar las causas de ciertas enfermedades. Asimismo, en el ámbito de seguridad, en los Estados Unidos de América, han dotado de sensores a las armas de los policías para conocer cuándo, dónde y por quién ha sido utilizada un arma. También, en el ámbito de la ganadería, se han diseñado pequeños objetos para monitorear la salud de las vacas y seguir sus movimientos, asegurando así la obtención de carne más saludable y abundante para el consumo humano [1].

En definitiva, existe una variedad de ejemplos de objetos dotados de sensores y conectados a Internet. Así, en el año 2011, la compañía de telecomunicaciones “Cisco” ya prevenía esta situación, y mediante un informe anunciaba que para el año 2020 aproximadamente se tendría 50.1 billones de “objetos” [2] conectados.

La conexión de “cosas” a Internet, es lo que se conoce como el “Internet de las Cosas” o también llamado “Internet of things” (IoT), un concepto que fue acuñado el año 1999 por Kevin Ashton [3] un investigador Británico que en aquellos años trabajaba en el Instituto Tecnológico de Massachusetts (MIT).

Desde la acuñación de este concepto y debido a su enorme potencial, el ámbito tecnológico puso su interés en el IoT. Por ello últimamente importantes compañías

internacionales han exhibido diferentes propuestas tecnológicas. Una de ellas ha sido Microsoft, que ha dado a conocer sus herramientas para la implementación del IoT: Azure y Windows 10 IoT Core. Por un lado Azure es un servicio de computación en la nube alojado en los centros de datos de Microsoft [4] y por otro, Windows 10 IoT Core es un Sistema Operativo diseñado para los pequeños Sistemas Empotrados [5].

Asimismo, la compañía Cisco System en su última versión de su simulador de redes "Packet Tracer 7.0" [6], introdujo el concepto del "Internet de las Cosas" (en inglés, Internet of Things, a partir de ahora IoT), es decir, entre los nuevos componentes de este simulador se pueden encontrar objetos cotidianos (puertas, lámparas, cafeteras, ventiladores, etc.) en el modelado de redes.

En la misma línea las compañías Samsung y Philips, lanzaron al mercado dispositivos electrónicos para el consumo del hogar. Samsung, presentó su línea "SmartThings" [7], un conjunto de dispositivos electrónicos (bombillas, altavoces, termostatos) y *hubs* que se conectan inalámbricamente. Philips, dentro de su línea "Philips Hue" [8], presentó un conjunto de bombillas led que pueden ser personalizadas desde un teléfono móvil.

Igualmente Google Inc. dio a conocer un ecosistema de tecnologías para la implementación de sistemas del IoT. Este ecosistema de tecnologías comprende: Android Things, Weave y Google Cloud Platform. Según la compañía, con el lanzamiento de estas tres tecnologías pretende facilitar el proceso de creación de productos del IoT. Es decir, Android Things se utilizaría para la gestión y ejecución de las aplicaciones de los Sistemas Empotrados y el protocolo Weave, para la comunicación entre los Sistemas Empotrados y la nube de Google. Google Cloud Platform para el procesamiento de los datos recolectados por los Sistemas Empotrados [9].

Así como existen varias propuestas tecnológicas para implementar productos del IoT en diferentes ámbitos, también existen aún problemas por solucionar en otros contextos, tales como: transporte y logística, cuidado de la salud, medioambiente, entornos inteligentes, futuristas, entre otros [10].

En el ejemplo concreto del ámbito de la salud, existen varios estudios médicos concernientes a los efectos medioambientales, específicamente a la temperatura y presión atmosférica sobre la salud de las personas.

Existen diferentes estudios médicos con respecto a los efectos medioambientales sobre la salud de las personas, así en primer lugar podemos citar a Castro [11] quien estudió las incidencias de las hemorragias subaracnoidea aneurismática y la influencia de los cambios de la presión atmosférica en los pacientes ingresados en el servicio de neurocirugía del Hospital Antonio Lenin Fonseca de Nicaragua. Los resultados que obtuvo en relación a la presión atmosférica y la temperatura a las cuales los pacientes se expusieron, son los siguientes: encontró una mayor incidencia de hemorragia subaracnoideas en aquellos que procedían de regiones donde se presentaban mayores cambios de la presión atmosférica y la temperatura, y en menor proporción observó en pacientes procedentes de regiones donde hubo una menor variación climatológica de la temperatura ambiental y presión atmosférica.

Del mismo modo, Gaona [12] evaluó la influencia de las variables meteorologías respecto a la movilidad respiratoria en niños de diferentes hogares distribuidos en la ciudad de Sao Paulo, Brasil, donde estudió los eventos de sibilancia (enfermedad respiratoria) relacionado con las condiciones de la calidad del aire y condiciones meteorológicas externas. Los resultados obtenidos giraron en torno a las variaciones de presión, las cuales generan frentes fríos y olas de calor, fenómenos estos susceptibles de ser los posibles factores de riesgo para el desarrollado de sibilancia.

Por su parte Olcina [13], analizó el efecto de las variaciones en la densidad de oxígeno -originadas por los cambios en las masas de aire presentes en la atmósfera-, sobre los ingresos hospitalarios por enfermedad cardiaca y pulmonares, registrados en la ciudad de Alicante (España). Los resultados que encontró fueron que una situación atmosférica que favorezca a una mayor o menor densidad de oxígeno no es causa directa del desarrollo dolencias, pero que si puede actuar como elemento favorecedor de su desarrollo en grupos de riesgo (personas mayores de 65 años) y personas con afección previa de dichas enfermedades. Destaca también el hecho de que las enfermedades cardíacas muestran una estacionalidad marcada, siendo los tres primeros y los tres últimos meses del año los que recogen mayor número de ingresos, coincidiendo con los meses más fríos.

De la misma forma, Paolasso [14] en su estudio de la Biometeorología, encontró factores relacionados con la composición de las masas de aire, la interacción entre la temperatura, presión atmosférica y humedad. En sus resultados, menciona que la baja presión atmosférica pudiera tener vinculación con la mala conducta de los escolares, y con algunos trastornos patológicos como el incremento de infartos cardíacos y ulceras sangrantes. También identificó posibles síntomas psíquicos tales como: el malestar

general, malhumor, tristeza, decaimiento o falta de fuerzas, irritabilidad y mutismo, que se producen cuando baja la presión atmosférica.

Como acabamos de mostrar, existen varias investigaciones relacionadas con los efectos medioambientales, específicamente la temperatura y la presión atmosférica sobre la salud de las personas. Siguiendo estas investigaciones sobre los distintos efectos medioambientales, hallamos un artículo publicado en la revista “La Vanguardia” [15] donde se explica que algunas personas sienten dolores en su cuerpo y cambian de conducta justo cuando hay un evento climático, no se sabe a ciencia exacta a qué se debe, sin embargo, estudios lo relacionan a los cambios climáticos tal como se comentó en párrafos anteriores.

Partiendo del anterior problema, entre las múltiples causas que puede traer este inconveniente, encontramos: fenómenos climatológicos inexplicables, contaminación del medioambiente, masas de aire caliente o frío, épocas del año, presencia de oxígeno en el aire, cambios constantes de temperie, habitad humano, factores genéticos, edad, alimentación, etc. [11], [12], [16], [13], [17], [14], [15].

Las consecuencias que pueden traer el problema de los dolores en el cuerpo y los cambios de conducta son muchas, entre las que hemos destacado las siguientes: disminución de la productividad, salud mental, cefaleas, equilibrio humorar, actividad nerviosa, aumento de enfermedades, enfermedades cardíacas, enfermedades pulmonares, enfermedades de presión arterial (alta y baja) e incluso la muerte [11], [12], [16], [13], [17], [14], [15].

En definitiva y como acabamos de analizar, existen muchas investigaciones asociadas a la influencia de los cambios de la temperatura y presión atmosférica en la salud de las personas, pero no existe una solución tecnológica que permita conocer las causas de esos efectos.

Las consideraciones expuestas nos llevan a plantear el siguiente interrogante:

¿Es posible implementar un sistema medioambiental de tal manera que permita conocer los efectos de la temperatura y presión atmosférica sobre la salud de las personas?

1.2 Objetivos

1.2.1 Objetivo general

Desarrollar un sistema medioambiental del Internet de las Cosas basados en tecnologías de Google, como apoyo para el estudio del efecto de la temperatura y la presión atmosférica sobre la salud de las personas.

1.2.2 Objetivos específicos

Como objetivos específicos para el trabajo de investigación proponemos:

1. Recopilar información relacionada con el “Internet de las Cosas”.
2. Explorar las tecnologías del “Internet de las Cosas” que se utilizan actualmente.
3. Estudiar los efectos de la temperatura y la presión atmosférica en el estado de salud de las personas.
4. Estudiar la tecnología Android Things de Google, utilizado para el desarrollo de Sistemas Empotrados.
5. Estudiar el protocolo de comunicación Weave de Google, utilizado para la comunicación de los dispositivos del “Internet de las Cosas”.
6. Estudiar la plataforma de computación en la nube Google Cloud Platform para la captura, procesamiento y almacenamiento de datos.
7. Implementar un Sistema Empotrado para la obtención de la temperatura y presión medioambiental.
8. Implementar el protocolo Weave para la comunicación de los dispositivos del “Internet de las Cosas”.
9. Implementar una arquitectura del “Internet de las Cosas” en la infraestructura de Google Cloud Platform.
10. Integrar Android Things, Weave y Google Cloud Platform.
11. Implementar una aplicación móvil para la evaluación del estado de salud de las personas.

1.3 Estructura del Trabajo

Para concretar una visión general del presente trabajo, la organización del mismo será la siguiente:

- Capítulo 1: Introducción. En este capítulo se dará a conocer las principales motivaciones para realizar el presente trabajo. Asimismo, se establecerán los objetivos que se lograrán con la investigación y finalmente se justificará nuestro estudio desde diferentes aspectos.
- Capítulo 2: Internet de las Cosas. En este capítulo se profundizará en el contexto sobre el que se desenvolverá nuestra investigación principal. Para ello primeramente, se puntualizarán sucesos históricos relacionados con IoT desde el año 1961 hasta el 2015. De igual forma se explicará el concepto del IoT y sus definiciones. También se describirán los elementos que componen el producto del IoT y se detallarán las diferentes propuestas de arquitecturas. Finalmente se puntualizarán las tecnologías utilizadas en cada elemento de un producto del IoT.
- Capítulo 3: Internet de las Cosas y tecnologías de Google. En este capítulo se describirá el ecosistema de tecnologías propuestas por Google para el Internet de las Cosas. Se explicará la arquitectura que ofrece Google para la integración de sus tecnologías y brevemente se puntualizará sobre Android Things, Weave y Google Cloud Platform describiendo sus características generales.
- Capítulo 4: Android Things. En este capítulo se aportará la definición de Android Things y se detallaran sus características. Se tratará además su evolución los requisitos de Software y Hardware para su implementación y se detallará cada componente de su arquitectura de Software. Android Things, al ser una version modificada de Android, se especificarán los paquetes de aplicaciones que no incluye. A continuación, se desglosarán los servicios de Google compatibles con Android Things, se harán referencia a aspectos de seguridad y se clasificarán los protocolos e interfaces utilizados en la comunicación con los periféricos de entrada y salida. Finalmente, se implementará una aplicación simple para mostrar el funcionamiento de Android Things.
- Capítulo 5: IoT basado en Weave. En este capítulo se definirá la plataforma de comunicación Weave y se describirán sus características generales. Se

enumerarán los requisitos de Hardware y se describirá el Software utilizado para su implementación. También, se mencionarán los aspectos de seguridad del protocolo Weave. Por último, se detallará la información obligatoria para la creación de un producto Weave.

- Capítulo 6: IoT basado en Google Cloud Platform. Un apartado donde se definirá Google Cloud Platform y se describirán sus características generales. Finalmente, se explicará de manera general todos los servicios de la plataforma agrupado en siete grupos: Procesamiento, Almacenamiento y base de datos, Redes, Big Data, Internet de las Cosas, Aprendizaje Automático, Identidad y Seguridad y Herramientas de Administración. Finalmente, se implementarán ejemplos básicos para exponer el funcionamiento de los servicios: Cloud Pub/Sub, Cloud Dataflow y BigQuery.
- Capítulo 7: Desarrollo de un sistema medioambiental basado en IoT. En este capítulo se implementará un sistema medioambiental del Internet de las Cosas utilizando tecnologías de Google. Se describirá el problema, la solución y el diseño del sistema medioambiental. En el diseño de la solución se propondrá una arquitectura de tres capas basada en la nube. Se describirá paso a paso la implementación de los componentes de Software y Hardware que se usarán en el desarrollo del sistema. También, se detallarán cada uno de los servicios de Google Cloud Platform utilizados en la solución.
- Capítulo 8: Conclusiones. En este último capítulo se obtendrán conclusiones con respecto al trabajo, que estarán relacionados con cada objetivo específico propuesto. Asimismo, se darán a conocer los problemas acaecidos durante la elaboración de este trabajo. Finalmente, para un adecuado enfoque del sistema medioambiental se mencionarán los trabajos futuros.

Capítulo 2

Internet de las Cosas

2.1 Antecedentes históricos

Siguiendo las investigaciones de Philip N. Howard [1] “Pax Technica: How the Internet of Things May Set Us Free or Lock Us UP” hallamos más de 134 eventos relacionados con el IoT, que abarcan una amplia gama de eventos sociales y de ingeniería relacionados con el desarrollo de dispositivos de redes y puede clasificarse por categorías tales como: drones y satélites, tipos de innovaciones, seguridad, tendencias en redes, cultura, políticas y normas técnicas.

De esos eventos, en esta sección se puntualizan solo los sucesos tecnológicos, foros y conferencias.

1961, el primer vínculo establecido con internet entre las universidades de UCLA y Stanford.

1962, orígenes de la computación en la nube, el psicólogo estadounidense y científico de computación Joseph Carl Robnett planteó su idea de una red intergaláctica.

1969, creación de ARPANET, una red de computadoras creada por encargo del departamento de defensa de los Estados Unidos.

1976, la primera tarjeta chip, el inventor francés Philip Moreno demostró que una tarjeta de plástico con un chip de computadora incrustado, podía utilizarse para pagos electrónicos.

1981, el primer computador portátil de propósito general, el investigador estadounidense Steve Mann diseño y construyó una computadora portátil multimedia con capacidades de comunicación inalámbrica. La computadora portátil aún no había sido inventada aún.

1982, los protocolos TCP/IP fueron establecidos como estándar. El departamento de defensa de los Estados Unidos estableció los protocolos TCP/IP como un estándar para todas las redes de computadoras militares.

1982, la primera máquina Coca Cola con conexión a internet.

1983, se patentó la tecnología de identificación por radio frecuencia (RFID).

1990, la tostadora conectada a internet, mediante conexiones TCP/IP -la tostadora se podía encender y apagar-.

1994, el primer teléfono inteligente del mundo -llamado Simon Personal Communicator-, fue introducido por IBM.

1992, se creó el primer protocolo M2M. MQTT (Message Queue Telemetry Transport), fue el primer protocolo para la comunicación máquina a máquina.

1999, el término "Internet de las Cosas" fue acuñado por el investigador británico Kevin Ashton, que en aquellos años trabajaba en el Instituto Tecnológico de Massachusetts (MIT).

2000, la primera ubicación de Geo cache, un juego de caza al aire libre usando dispositivos habilitados para GPS.

2000, el primer refrigerador conectado al internet fue presentado por LG Electronics.

2002, introducción de la tecnología NFC. Sony y Philips anunciaron su cooperación para crear la tecnología NFC.

2005, el primer informe de las Naciones Unidas sobre el "Internet de las Cosas". El organismo especializado de las Naciones Unidas para las Tecnologías de Información y Comunicación(TIC) y la Unión Internacional de Telecomunicaciones (UIT) publicaron un primer informe sobre el Internet de las Cosas.

2006, introducción de la tecnología Bluetooth. Nokia presentó la tecnología Bluetooth bajo el nombre de "Wibree".

2007, fundación de la Organización Europea de Investigación de Internet de las Cosas. La Dirección General de la Sociedad de la Información y los Medios de Comunicación de la Comisión Europea, puso en marcha el foro Networked Enterprise y RFID que más tarde se convertiría en el Clúster Europeo de Investigación sobre el Internet de las Cosas (IERC).

2008, la primera conferencia sobre el Internet de las Cosas. Organizada en Zúrich, fue

el primer evento europeo que reunió a investigadores de todo el mundo.

2009, las primeras aplicaciones de nube basadas en la computadora. Google lanzó la primera aplicación empresarial basada en un navegador a gran escala.

2009, el motor de búsqueda de Internet de las Cosas. SHODAN es un buscador de dispositivos conectados a internet -muchos de ellos con acceso público-.

2012, lanzamiento mundial del protocolo IPv6. La versión más reciente del protocolo de Internet IPv6 extiende el número de dirección IP de 2^{32} a 2^{128} nuevas direcciones IP.

2014, fundación del Consorcio Industrial de Internet de las Cosas (IIC). El propósito de IIC es reunir a la industria en conjunto, los grupos de tecnología, la academia y las instituciones gubernamentales para avanzar en el desarrollo de la accesibilidad de los dispositivos basados en Internet.

2014, segundo foro mundial de Internet de las Cosas. Cisco en colaboración con los miembros del comité directivo de la industria, organizaron el segundo foro mundial del Internet de las Cosas en Chicago – Estados Unidos.

2015, Google desarrolla su sistema operativo para el Internet de las Cosas. Brillo es un sistema operativo para dispositivos de baja potencia del Internet de las Cosas.

2015, conferencia mundial y exposición de Internet de las Cosas. La 2^a Conferencia anual de Internet de las Cosas fue celebrada en San Francisco (Estados Unidos).

2.2 Internet de las Cosas

El término “Internet de las Cosas” fue acuñado en 1999 por Kevin Ashton un investigador británico que en aquellos años trabajaba en el Instituto Tecnológico de Massachusetts(MIT) como fundador y director del grupo de Investigación Auto-ID, lugar donde se efectuaban investigaciones en el campo de la Identificación por Radio Frecuencia(RFID) y sobre tecnologías de sensores.

Kevin Ashton propuso el término de “Internet de las Cosas” como parte de una presentación en Procter & Gamble(P&G)¹. Este autor, también explicó entonces que las computadoras de hoy y por lo tanto Internet, son casi totalmente dependientes de los seres humanos para generar información, propuso potenciar las computadoras con sus propios medios de recopilación de información para que pueden hablar, ver y oír el mundo por sí mismos [18].

2.3 Definiciones

Actualmente, el término “Internet de las Cosas” tiene varias definiciones con perspectivas “orientadas a Internet” o “orientadas a las cosas” o “orientadas a la semántica” dependiendo de las finalidades de cada organización interesada [19].

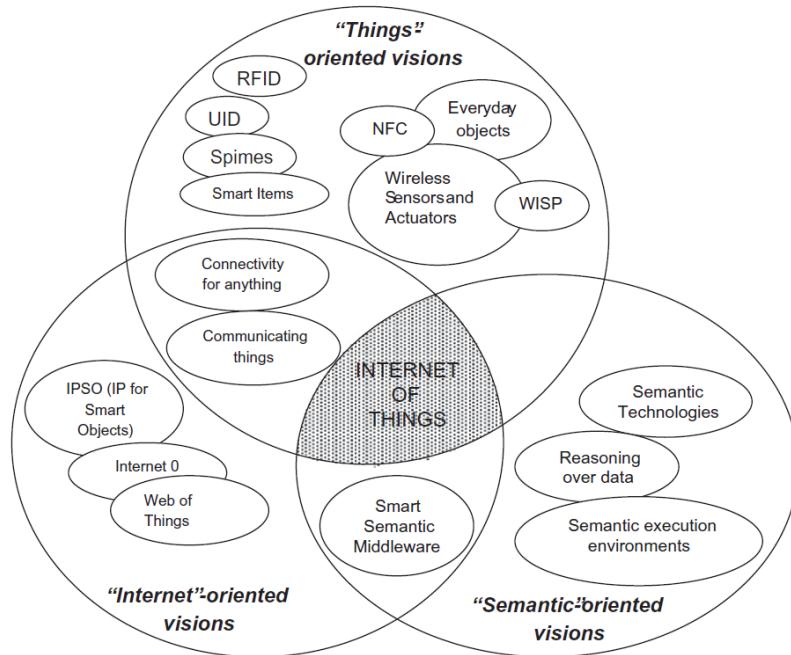


Figura 1: Visión del IoT [10].

¹ Procter & Gamble es una empresa estadounidense multinacional de bienes de consumo con sede en Cincinnati, Ohio.

Por ejemplo, el grupo RFID define el Internet de las Cosas como:

“La red mundial de objetos interconectados, exclusivamente direccionables a base de protocolos de comunicación estándar.”

Según el Cluster Europeo de Investigación sobre el Internet de las Cosas (IERC – European Research Cluster on the Internet of Things):

“Las ‘Cosas’ son participantes activos en la empresa, la información y los procesos sociales en los que están habilitados para interactuar y comunicarse entre ellos y con el medioambiente mediante el intercambio de datos e información obtenidos por el medioambiente, mientras que reaccionan de forma autónoma a los conocimientos del mundo real/físico e influenciarlo por los procesos que desencadenan las acciones y crear servicios con o sin intervención humana directa.”

Según Forrester, un entorno inteligente:

“Utiliza tecnologías de información y comunicaciones para que los componentes críticos de la infraestructura y servicios de la administración de una ciudad, la educación, la salud, la seguridad pública, los bienes raíces, el transporte y los servicios públicos sean más conscientes, interactivos y eficientes.”

Como acabamos de comprobar, no es fácil encontrar una definición completa del IoT. La definición a menudo depende de la visión particular de la entidad que la emite con respecto a los activos del Internet de las Cosas que se consideren relevantes. En otras palabras, muchas definiciones están sesgadas hacia los activos que un proponente específico quiere enfatizar.

2.4 Elementos del Internet de las Cosas

Para comprender el concepto de Internet de las Cosas en las siguientes secciones se describen seis elementos necesarios para proporcionar la funcionalidad del IoT [20].



Figura 2: Elementos del IoT [20].

2.4.1 Identificación

El elemento de identificación es crucial para identificar los objetos del IoT. Dentro de este elemento se hace referencia a los métodos de nombramiento y direccionamiento de los objetos del IoT. Muchos métodos de nombramiento están disponibles para el IoT tales como etiquetas uCode², EPC³, RFID, NFC, QR, códigos de barra entre otras formas. Además, el direccionamiento de los objetos IoT es fundamental para diferenciar entre nombre del objeto y su dirección del objeto dentro de una red de comunicaciones. El nombre de objeto se refiere a un nombre -como por ejemplo “SEN1”- para un sensor de temperatura particular, y la dirección del objeto se refiere a su dirección dentro de una red de comunicaciones. Además, los métodos de direccionamiento de los objetos IoT incluyen IPv6 e IPv4. La distinción entre el nombre y la dirección del objeto es imprescindible, los métodos de nombramiento no son globalmente únicos y el direccionamiento ayuda a identificar el objeto de forma única. Por último, los métodos de identificación se utilizan para proporcionar una identidad clara para cada objeto dentro de la red.

2.4.2 Detección

El componente de detección tiene la finalidad de recopilar datos de los objetos dentro de la red y enviarlos a un almacén de datos que puede ser una base datos o un servicio en la nube. Los datos pueden ser capturados por un dispositivo móvil, dispositivo wearable, sensores, o cualquier otro dispositivo que permita recolectar datos.

² uCode es un código para identificar objetos individuales.

³ El código electrónico de producto (EPC) es un código estandarizado, al igual que el código de barras.

Las computadoras de placa única (Arduino, Raspberry PI, BeagleBone, etc.) integrados con sensores y funcionalidad TCP/ IP, se usan de forma típica para realizar productos IoT.

2.4.3 Comunicación

El componente de comunicación permite la conexión de objetos heterogéneos. Normalmente los nodos IoT operan con un bajo consumo de energía, en entornos de enlaces de comunicación ruidosos y con altos niveles de perdida de información; por lo que requiere protocolos de comunicación especializados. Los protocolos de comunicación utilizados para el IoT comúnmente son: Wifi, Bluetooth, Zigbee y Z-wave entre otros.

2.4.4 Computación

El componente de computación tiene la finalidad de procesar los datos obtenidos en el componente de detección, el procesamiento puede realizarse por medio de Hardware o Software. Las unidades de procesamiento tales como microcontroladores, microproycesadores y aplicaciones de Software representan el cerebro y la capacidad computación del IoT.

Por su parte también se desarrollaron varias plataformas de Hardware para ejecutar aplicaciones de IoT como Arduino, Raspebery PI, Intel Galileo, entre otros.

Además, muchas plataformas de Software se utilizan para proporcionar funcionalidades IoT. Entre estas plataformas están los sistemas operativos ligeros que son vitales para entornos IoT ya que funcionan durante todo el tiempo de activación de un dispositivo.

Las plataformas en la nube forman otra parte computacional importante del IoT. Estas plataformas proporcionan facilidades para que los objetos envíen sus datos a la nube y así se procesen grandes volúmenes de datos en tiempo real; con ello los usuarios finales se beneficiarán del conocimiento resultante de los grandes volúmenes de datos recolectados.

En conclusión, dependiendo del tamaño de los datos obtenidos en el componente de detección, el procesamiento de los mismos se puede realizar en el objeto o fuera del objeto.

2.4.5 Servicios

En general, los servicios de IoT se pueden clasificar en categorías, tal como se propone y describe en [21]. Estas categorías son:

- Servicios relacionados con la identidad
- Servicios de agregación de información
- Servicios de colaboración
- Servicios ubíquos.

Servicios relacionados con la identidad

Los servicios relacionados con la identidad son los más simples, pero tal vez uno de los más importantes servicios que se deben proporcionar al IoT. La aplicación de un servicio relacionado con la identidad proporciona al desarrollador la información vital acerca de todos los dispositivos. La tecnología más importante utilizada en los servicios relacionados con la identidad, es RFID. Algunos ejemplos de aplicaciones que utilizan un servicio relacionado con la identidad pueden ser: la producción, el transporte, cadena de suministro, etc.

Servicios de agregación de información

Los servicios de agregación de información incorporan servicios relacionados con la identidad, junto con otros componentes tales como redes de sensores inalámbricos y acceso Gateway. El servicio de agregación de información es el responsable de proporcionar a la aplicación toda la información que se recoge y son útiles en situaciones de monitorización. Algunos ejemplos de aplicaciones que utilizan un servicio de agregación de información pueden ser: monitorización de la energía de la casa, producción de la agricultura en invernaderos, supervisión de datos fisiológicos de pacientes, etc.

Servicios de colaboración

La principal diferencia entre los servicios de agregación de información y servicios de colaboración, radica en el uso de los datos recogidos para tomar decisiones y realizar acciones.

Servicios ubicuos

Los servicios ubicuos son el objetivo final del IoT, ya que proporcionan un acceso completo y el control de todo lo que nos rodea, ya sea a través de una computadora o un teléfono móvil o alguna otra cosa. Los servicios ubicuos, proveen servicios colaborativos en cualquier momento que sean necesarios a cualquiera que los necesite en cualquier lugar.

2.4.6 Semántica

El componente de la semántica representa el cerebro del IoT. Se refiere a la capacidad de extraer conocimiento inteligente por diferentes máquinas a fin proporcionar los servicios necesarios. La extracción del conocimiento incluye el descubrimiento, utilización de los recursos y modelado de la información. Además de todo ello, también incorpora el análisis de datos para dar sentido a la decisión correcta de proporcionar el servicio exacto. Las tecnologías comúnmente utilizadas son: marco de descripción de recursos (RDF), lenguaje de ontología web (OWL), formato de intercambio eficiente de XML (EXI), etc.

2.5 Arquitectura

En el ámbito de la investigación, hay una variedad de propuestas arquitectónicas para el IoT. Existen arquitecturas simples de tres capas hasta arquitecturas complejas de n-capas. En esta sección se explicarán cuatro arquitecturas utilizadas en el IoT:

- Arquitectura de 3 capas
- Arquitectura basada en middleware
- Arquitectura basada en SOA
- Arquitectura de 5 capas

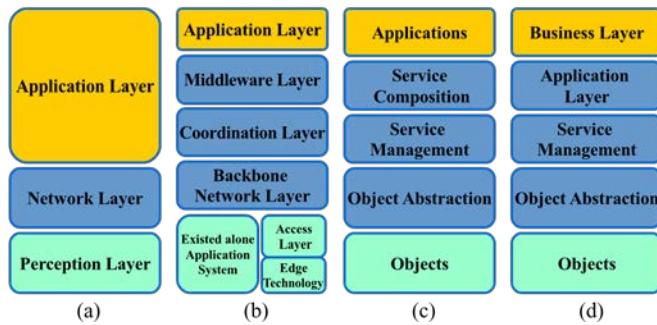


Figura 3: Propuestas de arquitecturas para el IoT [20].

2.5.1 Arquitectura de 3 capas

Capa de percepción

La capa de percepción tiene la principal tarea de percibir las propiedades físicas de los objetos que nos rodean y que forman parte del IoT. Este proceso de percepción se basa en varias tecnologías de detección como: RFID, NFC, GPS, sensores, o cualquier otro dispositivo que permita recopilar datos. Además, esta capa se encarga de convertir la información en señales digitales. Sin embargo, algunos objetos no pueden percibir datos directamente. En consecuencia, los microcontroladores deben ser añadidos a estos objetos para mejorarlo con capacidades de detección e incluso de procesamiento. De hecho, las nanotecnologías y los Sistemas Empotrados desempeñan un papel importante en la capa de percepción. El primero hará microcontroladores lo suficientemente pequeños para ser implantados en los objetos usados en nuestra vida cotidiana, mientras que el segundo los mejorará con capacidades de procesamiento que son requeridas por cualquier aplicación futura [20].

Capa de red

La capa de red es la responsable de procesar los datos recibidos de la capa de percepción. Además, se encarga de transmitir datos a la capa de aplicación a través de varias tecnologías de red, como redes inalámbricas/cableadas y redes de área local (LAN). Los medios principales para la transmisión incluyen 3G/4G, Wifi, Bluetooth, Zigbee, tecnología infrarroja, etc. Enormes cantidades de datos serán transportados por la red. Por lo tanto, es crucial proporcionar un middleware para almacenar y procesar esta enorme cantidad de datos. Para alcanzar este objetivo, la computación

en la nube es la tecnología primaria en esta capa. Esta tecnología proporciona una interfaz fiable y dinámica a través de la cual se pueden almacenar y procesar datos [20].

Capa de aplicación

La capa de aplicación utiliza los datos procesados por la capa anterior. De hecho, esta capa constituye el “front-end”⁴ de toda la arquitectura IoT, a través de la cual se explotará el potencial de IoT. Por otro lado, esta capa proporciona las herramientas requeridas para que los desarrolladores realicen la visión de IoT. En esta visión, el rango de las posibles aplicaciones es asombroso, por ejemplo: casas inteligentes, ciudades inteligentes, atención médica, cadena de suministros, agricultura, medioambiente, seguridad, etc. [20].

2.5.2 Arquitectura basada en middleware

Capa de tecnología de borde – Capa acceso – Sistema de aplicación existente

La capa de tecnología de borde es una capa de detección, que se compone de Sistemas Empotrados y dispositivos de Hardware. Esta capa tiene la responsabilidad de percibir los datos del entorno.

La capa de acceso es una puerta de enlace responsable de gestionar los datos y se utiliza para la publicación y la suscripción a los servicios y la comunicación entre plataformas.

La capa de aplicación existente, es la estandarización del Internet de Cosas para habilitar la interoperabilidad, no afecta a la aplicación existente y el nuevo objeto de implementación [22].

Capa de red troncal

Una red troncal es una red utilizada para interconectar otras redes. Suelen ser de alta capacidad y permiten un mayor rendimiento de las conexiones.

⁴ Los términos “front-end” junto al “back-end” proviene del ámbito web. El término “front-end” se refiere a la parte de Software que interactúa con los usuarios y el “back-end” a la parte que procesa la entrada desde el “front-end”.

La capa de red realiza las funciones básicas de datos y transmisión de información, así como la interconexión de sistemas y plataformas [23] [24].

Capa de coordinación

La capa de coordinación es la responsable de procesar la estructura de paquetes de diferentes redes y montarlos en una estructura unificada que puede ser procesada y volverlo a montar en una estructura unificada que pueda ser identificada y procesada por cada aplicación [23] [24].

Capa de middleware

La capa de middleware es una capa de Software interpuesta entre los niveles tecnológicos y aplicación. El middleware abstrae los detalles de las diferentes tecnologías adoptadas por las capas inferiores excluyendo al programador del conocimiento exacto del conjunto de tecnologías. Al ocultar los detalles de las diferentes tecnologías, proporciona una interfaz de programación de aplicaciones para la comunicación, administración de datos, computación, seguridad y privacidad [23] [24].

Capa de aplicación

La capa de aplicación es la que se encuentra en la parte superior de la arquitectura, exportando todas las funcionalidades del sistema al usuario final, es la que posibilita el desarrollo de un gran número de aplicaciones [23] [24].

2.5.3 Arquitectura basada en SOA

Objetos

Está integrada con los objetos de Hardware disponibles en la red que detectan el estado de las cosas. En la capa de objetos, los sistemas inteligentes mediante etiquetas RFID o sensores, son capaces de detectar automáticamente el medioambiente y el intercambio de datos entre los dispositivos. Los objetos de esta capa deben tener una dirección digital única, lo que permite rastrear al objeto en el dominio digital, posibilitando cumplir con la expectativa de IoT de ser una red física interconectada en todo el mundo, en el que las cosas están conectadas a la perfección y se pueden controlar de forma remota [10].

Abstracción de objetos

El IoT se basa en un conjunto amplio y heterogéneo de objetos, cada uno proporcionando funciones específicas accesibles a través de su propio dialecto. Por tanto, existe la necesidad de una capa de abstracción capaz de armonizar el acceso a los diferentes dispositivos con un lenguaje y un procedimiento común.

Por consiguiente, existe la necesidad de introducir una capa de envoltura, que consiste en dos subcapas la interfaz y las comunicaciones. La subcapa de interfaz, expone métodos disponibles a través de una interfaz de servicios web estándar y es responsable de la gestión de todas las operaciones de mensajería entrantes y salientes involucradas en la comunicación con el mundo exterior. La subcapa de comunicación, implementa la lógica detrás de los métodos de servicio web y traduce estos métodos en un conjunto de comandos específicos del dispositivo para comunicarse con los objetos del mundo real [10].

Gestión de servicio

En esta capa se crean y gestionan los servicios requeridos por los usuarios o aplicaciones de Software. La capa de servicio se basa en la tecnología de middleware, la cuál es fundamental para consumir servicios y para la ejecución de aplicaciones de IoT, donde las plataformas de Hardware y Software pueden ser reutilizables. Es una de las capas de operación crítica de la arquitectura, que funciona en modo bidireccional. Esta capa opera como interfaz entre la capa de objetos, y la capa de aplicación. Es responsable de funciones como la gestión de dispositivos, gestión de información, filtrado de datos, agregación de datos, análisis semántico, y descubrimiento de información [10].

Composición de servicio

La capa de composición de servicio proporciona las funcionalidades para la composición de los servicios individuales ofrecidos por los objetos en red para crear aplicaciones específicas. Sobre esta capa no hay noción de dispositivos y los únicos activos son los servicios. El rol importante de esta capa es tener un repositorio de todas las instancias de servicios para construir servicios compuestos. La lógica detrás de la creación y la gestión de servicios complejos, se puede expresar en términos de flujos de trabajo de procesos de negocio, utilizando lenguajes de flujo de trabajo como

BPEL⁵ y Jolie⁶ [10].

Aplicación

Es la capa responsable de la entrega de las aplicaciones a los diferentes usuarios de IoT. El desarrollo de aplicaciones en IoT se ha centrado en las áreas de: salud, agricultura, transporte, ciudades inteligentes, casas inteligentes, sistemas complejos para la toma de decisiones, gestión de uso de agua, etc. [10].

2.5.4 Arquitectura de 5 capas

Objetos

La capa de objetos, representa los sensores físicos de la IoT que apuntan a recopilar y procesar información. Esta capa incluye sensores y actuadores para realizar diferentes funcionalidades tales como: localización, temperatura, peso, movimiento, vibración, aceleración, humedad, etc. Los mecanismos estandarizados de “plug and play”⁷ deben ser utilizados por la capa de percepción para configurar objetos heterogéneos. La capa de percepción transfiere los datos a la capa de abstracción de objetos a través de canales seguros. Los grandes datos creados por el IoT se inician en esta capa [20].

Abstracción de objetos

La capa de abstracción de objetos transfiere los datos producidos por la capa de objetos a la capa de gestión de servicios a través de canales seguros. Los datos pueden ser transferidos a través de diversas tecnologías como: RFID, 3G/4G, GSM, Wifi, Bluetooth Baja Energía, infrarrojos, ZigBee, etc. Además, otras funciones como la computación en nube y los procesos de gestión de datos se manejan en esta capa [20].

Gestión de servicio

⁵ BPEL es un lenguaje de orquestación de servicios web. Es un lenguaje basado en XML que soporta las tecnologías de servicios Web (Incluyendo SOAP, WSDL, UDDI, WS-Reliable Messaging, WS-Addressing, WS-Coordination y WS-Transaction)

⁶ Jolie es un lenguaje de programación para el desarrollo de aplicaciones distribuidas sobre la base de micro servicios.

⁷ Plug and Play, es la tecnología que permite a un dispositivo informático ser conectado a una computadora sin tener que configurar, mediante Software específico proporcionado por el fabricante.

La capa de gestión de servicio o Middleware permite a los programadores de aplicaciones IoT trabajar con objetos heterogéneos sin tener en cuenta una plataforma Hardware específica. Además, esta capa procesa los datos recibidos, toma decisiones y entrega los servicios requeridos a través de los protocolos de red [20].

Capa de aplicación

La capa de aplicación provee los servicios solicitados por los clientes. Por ejemplo, la capa de aplicación puede proporcionar mediciones de temperatura y humedad del aire al cliente que solicita dichos datos. La importancia de esta capa para el IoT radica en que tiene la capacidad de proporcionar servicios inteligentes de alta calidad para satisfacer las necesidades de los clientes. La capa de aplicación abarca numerosos mercados como: casas inteligentes, la construcción inteligente, el transporte, la automatización industrial y la salud inteligente [20].

Capa de negocio

La capa de negocio gestiona las actividades y los servicios generales del sistema IoT, es la encargada de producir análisis e informes de alto nivel. Las responsabilidades de esta capa son: construir un modelo de negocio, gráficos, diagramas de flujo, etc. Todos ellos basados en los datos recibidos de la capa de aplicación. La capa de negocio hace posible soportar procesos de toma de decisiones basados en el análisis de Big Data. Además, el control y la gestión de las cuatro capas inferiores se consigue en esta capa, responsable asimismo de comparar la producción de cada capa con la producción esperada para mejorar los servicios y mantener la privacidad de los usuarios. Esta capa se aloja en dispositivos potentes debido a sus complejas y enormes necesidades computacionales [20].

2.6 Tecnologías

En esta sección se describirán las tecnologías del Internet de las cosas más utilizadas para su implementación [25]. Estas tecnologías estarán agrupadas según la Tabla 1:

Tabla 1: Clasificación de tecnologías por elementos del IoT.

Elementos del IoT		Tecnologías
Identificación	Nombramiento	EPC, uCode
	Direccionamiento	IPv4, IPv6
Detección		Teléfonos inteligentes, Werable, Sensores, Actuadores, Sistemas Empotrados, Etiquetas RFID
Comunicación	Tecnologías de comunicación	Wifi, Bluetooth, Zigbee, SigFox, LoRa, NB-IoT
	Protocolos de comunicación	CoAP, MQTT, XMPP, AMQP, HTTP
Computación	Hardware	Raspberry PI, Arduino, Intel Edison, Intel Joule, versiones NXP, Phidgets, BeagleBone, Nanode
	Software	Contiki, Mbed OS, TynnyOS, Micrium OS, RIOT, Brillo, Xively, Kaa, IBM Bluemix, Carriots, Nimbts

2.6.1 Identificación

Nombramiento

EPC

El EPC (Electronic Product Code) es un estándar de codificación que se utiliza mundialmente para la identificación por radiofrecuencia (RFID). El código EPC contiene información de productos y localizaciones. También pueden incluir información específica como la información de lote, fechas de caducidad para la trazabilidad y la seguridad alimentaria. La información de este código se almacena en una etiqueta de radiofrecuencia, que transmite los datos mediante una señal emitida por un lector especial [26].

uCode

Un uCode (Ubiquitous Code) es un número de identificación que se puede usar para identificar “objetos” de manera única. Cuando un uCode se emite para un “objeto”, el mismo se almacena en una etiqueta (esto se denomina uCode tag) similar a un código de barras o una etiqueta RFID. La información de los “objetos” se almacenan en la base de datos de la infraestructura de uCode Center. Esta información de los objetos identificados se puede recuperar de las bases de datos mediante el uso de la

uCode como clave. El uCode es un número de identificación, es esencial para garantizar la singularidad de las “cosas” identificados [27].

Direccionamiento

IPv4

IPv4 es la cuarta versión del Protocolo de Internet(IP) y es la primera versión del protocolo que se implementa ampliamente. De hecho es la versión actual del Protocolo de Internet, (que es el sistema de identificación que utiliza Internet para enviar información entre dispositivos). Utiliza un direccionamiento de 32 bits. Su formato de direcciones es decimal. Este sistema asigna una dirección de cuatro octetos (cada uno de los cuales está comprendido entre 0 y 255) a cada dispositivo. IPv4 solo permite 2^{32} direcciones (aproximadamente 4.000 millones de direcciones) [28].

IPv6

IPv6 significa protocolo de Internet versión 6, es la nueva versión del protocolo de Internet y amplía el número de direcciones disponibles. Utiliza un direccionamiento de 128 bits. Su formato de direcciones es hexadecimal. Este sistema asigna una dirección de 8 segmentos (cada segmento comprende 16 bits representado con números hexadecimales) a cada dispositivo. IPv6 permite 2^{128} direcciones (una cantidad prácticamente ilimitada de 340 sextillones de direcciones). La migración a IPv6 permite que Internet siga creciendo, así como el desarrollo de servicios nuevos e innovadores, debido a que es mayor el número de dispositivos que se pueden conectar a Internet [28].

2.6.2 Detección

Teléfonos inteligentes

Los Teléfonos Inteligentes, también conocidos como Smartphone, se pueden definir como teléfonos celulares que ofrecen prestaciones similares a las que brinda una computadora y que destaca por su conectividad. El teléfono inteligente cuenta con todas las funciones básicas del teléfono celular (permite realizar llamadas telefónicas, enviar mensajes de texto, etc.) y le agrega características avanzadas (sensores, conexión a Internet, capacidades multimedia, pantalla táctil) [29].

Wearables

Dispositivos electrónicos diseñados para llevarlos siempre puestos. Esta palabra proviene del inglés, y hace referencia a accesorios tecnológicos que una persona puede llevar puestos [30].

Sensores

Los sensores son dispositivos electrónicos compuestos de células sensibles que son capaces de medir parámetros físicos como la fluctuación de la luz, utilizando una fotorresistencia; la temperatura, usando un termistor; para detectar llamas, sonidos, movimientos o cualquier otra fluctuación en el medioambiente [31].

Actuadores

Un Actuador es un dispositivo mecánico cuya función es proporcionar fuerza para mover o “actuar” otro dispositivo mecánico. La fuerza que provoca el actuador procede de tres fuentes posibles: presión neumática, presión hidráulica y fuerza motriz eléctrica [32].

Etiqueta RFID

La etiqueta RFID es un dispositivo pequeño, que puede ser adherido o incorporado a un producto, animal o persona. Contienen antenas para permitirles recibir y responder a peticiones por radiofrecuencia desde un emisor/receptor RFID [33].

2.6.3 Comunicación

Tecnologías de comunicación

Wifi

Wifi es una tecnología de red inalámbrica que permite que las computadoras y otros dispositivos se comuniquen a través de una señal inalámbrica. Comprende un conjunto de estándares para redes inalámbricas basados en las especificaciones IEEE 802.11, lo cual asegura la compatibilidad e interoperabilidad en los equipos certificados bajo esta denominación. Su alcance es de 100 metros y su velocidad de transmisión de datos es de 11 Mbps [34].

Bluetooth

Bluetooth, también conocido como el estándar IEEE 802.15.1, es una tecnología inalámbrica que permite la comunicación entre dispositivos compatibles con Bluetooth. Se utiliza para conexiones de corto alcance entre las computadoras de escritorio y portátiles, PDAs, cámaras digitales, escáneres, teléfonos móviles, impresoras, etc. Bluetooth utiliza una frecuencia estándar de 2.4 GHz para que todos los dispositivos con Bluetooth sean compatibles entre sí. Su alcance es limitado a 10 metros y su velocidad de transmisión datos es de 1 Mbps [34].

Zigbee

Zigbee a través de IEEE 802.15.4, define una especificación de red de malla para redes inalámbricas de área local (WLAN) de baja potencia que cubren un área grande. ZigBee ha sido diseñado para dispositivos simples que consumen poca energía, que operan en un espacio personal y que no tienen la necesidad de enviar grandes cantidades de datos. Su alcance es de 70 a 300 metros y su velocidad de transmisión datos es de 250 kbps [34].

SigFox

SigFox es una solución de conectividad celular mundial para el Internet de las Cosas, pensada para comunicaciones de baja velocidad, que permite reducir el consumo de energía para los dispositivos conectados. La solución de SigFox se basa en una infraestructura de antenas y estaciones base totalmente independientes de las redes existentes. SigFox es una solución de conectividad que se concentra en los dispositivos de baja velocidad. Además es compatible con tecnologías Bluetooth, 2G/3G/4G y Wifi. En SigFox se pueden enviar entre 0 y 140 mensajes por día y cada mensaje puede contener hasta 12 bytes de datos reales de carga útil. Su velocidad de transmisión de datos es de 100 bps [35].

LoRa

LoRaWAN (acrónimo de Long Range Wide Área Network) es una especificación LPWAN (Low Power Wide Area Network), o expresado de forma más sencilla: red de largo alcance y bajo consumo. Consuma tan poco que permite a los dispositivos funcionar durante años con batería. Tiene un gran alcance que permite llegar a equipos de difícil acceso. Su velocidad de transmisión de datos es de 0.3 kbps a 50 kbps [36].

NB-IoT

NarrowBand IoT o CAT NB1 es una red de largo alcance y bajo consumo, desarrollado para permitir a una amplia gama de dispositivos, estar conectados con bandas de telecomunicación celulares. NB-IoT es la iniciativa por el 3GPP, el organismo de normalización que escribe los estándares celulares, para hacer frente a las necesidades de los dispositivos de muy baja velocidad de datos que necesitan conectarse a redes de telefonía móvil. Su velocidad de transmisión es de 250 kbps [37].

Protocolos de comunicación

CoAP

El Protocolo de Aplicación Restringida, es un protocolo de la capa de aplicación. CoAP define un protocolo de transferencia web basado en REST sobre las funcionalidades HTTP. Está vinculado a UDP. CoAP modifica algunas funciones HTTP para cumplir con los requisitos del IoT, como bajo consumo de energía y funcionamiento en presencia de enlaces con pérdidas y ruidos. Un mensaje CoAP típico puede estar entre 10 y 20 bytes. Dado que CoAP se ha diseñado basado en REST, la conversión entre estos dos protocolos es sencilla [38].

MQTT

Es un protocolo de mensajería que se introdujo en 1999 y fue estandarizado el 2013 en OASIS. MQTT apunta a la conexión de dispositivos y redes integradas con aplicaciones y middleware. La operación de conexión utiliza un mecanismo de encaminamiento (uno a uno, uno a mucho, muchos a muchos) lo que permite a MQTT ser un protocolo de conexión para la IoT. MQTT utiliza el patrón publicación/suscripción para proporcionar la flexibilidad de transición y simplicidad de implementación. MQTT es adecuado para dispositivos con restricciones de recursos que utilizan enlaces no fiables o de bajo ancho de banda. MQTT se construye en la parte superior del protocolo TCP. Existen dos especificaciones principales para MQTT: MQTT v3.1 y MQTT-SN V1.2. Este último se definió específicamente para las redes de sensores y define un mapeo UDP.

MQTT está formado por tres componentes: Publicador, Suscriptor y Bróker. A un dispositivo que necesita obtener datos se lo conoce como Suscriptor y a los dispositivos

que necesiten transmitir información se los conoce como Publicador. El Bróker es el componente que gestiona los temas y las suscripciones de los dispositivos. Es el intermediario que recolecta y entrega información a las entidades interesadas [38].

XMPP

El Protocolo Extensible de Mensajería y Presencia, es un protocolo abierto y extensible basado en XML, originalmente ideado para mensajería instantánea. Es un protocolo estándar que se utiliza para chat, llamadas de voz, llamadas de video y tele presencia⁸. XMPP fue desarrollado por la comunidad de código abierto Jabber para soportar un protocolo de mensajería abierto y seguro. Este también permite a los usuarios comunicarse entre sí mediante el envío de mensajes instantáneos en internet, independientemente del sistema operativo que utilicen. XMPP conecta un cliente a un servidor usando una stanza XML. Por su parte, una stanza XML representa un fragmento de código que se divide en tres componentes: mensaje, presencia e IQ. El fragmento de mensaje, rellena los campos subject y body con el título del mensaje y el contenido respectivamente. El fragmento presencia, muestra y notifica a los clientes de las actualizaciones de estado según se autorizan. El fragmento IQ, empareja los remitentes y receptores de mensajes [38].

AMQP

Es un protocolo de capa de aplicación estándar abierta para el IoT que se centra en entornos orientados a mensajes. AMQP requiere un protocolo de transporte de confianza como TCP para intercambiar mensajes. Las comunicaciones son controladas por dos componentes principales, estos son: Intercambios y Colas de Mensajes. Los Intercambios se utilizan para enrutar los mensajes a las colas apropiadas, los mensajes pueden almacenarse en las colas de mensaje y a continuación, enviarse a los receptores. AMQP también es compatible con el modelo de comunicaciones publicación/suscripción [38].

HTTP

El protocolo de transferencia de hipertexto (Hypertext Transfer Protocol o HTTP) es el protocolo de comunicación que permite las transferencias de información a través

⁸ Telepresencia, de acuerdo a una presentación de Cisco acerca de este tema, la Telepresencia se enfoca en la experiencia del usuario. Simula la interacción de las personas como si estuvieran en la misma sala de reuniones, tiene especial cuidado en la domótica, la iluminación, el mobiliario y otros detalles. En cambio, la Video Conferencia solo se enfoca en la comunicación a través de video.

de Internet u otra red informática. Es un protocolo de intercambio de información entre el cliente y servidor y está basado en el esquema petición/respuesta. HTTP es un protocolo sin estado, es decir, no guarda ninguna información sobre conexiones anteriores [39].

2.6.4 Computación

Hardware

Raspberry Pi

Es una mini computadora con el tamaño de una tarjeta de crédito de bajo costo. Fue desarrollado en Reino Unido por la Fundación Raspberry PI, con el objetivo principal de estimular la enseñanza de ciencias de la computación en las escuelas. Al ser una mini computadora se puede utilizar para desarrollar cosas bastante más complejas que con Arduino y puede utilizar lenguajes de programación de alto nivel como Python, C++ y Java [40].

Arduino

Arduino es una plataforma abierta y versátil para el desarrollo de productos electrónicos muy enfocado a un público no experto. Al ser Hardware libre, todos sus diseños son abiertos y pueden utilizarse e incluso mejorarse. Su principal característica es que son muy fáciles de programar y tienen un bajo costo [41].

Intel Edison

Intel Edison es un módulo de computación de Intel pequeño y potente. El chip incluye: una CPU, memoria, almacenamiento y doble banda Wifi y Bluetooth. El modulo se puede montar en una placa de expansión lo que permite la creación rápida de prototipos y aplicaciones de Internet de las Cosas [42].

Intel Joule

Intel Joule es una placa computacional o micro computador, el módulo de Intel Joule es un microcontrolador de bajo consumo, está pensado para utilizar en el desarrollo de visión computacional, robótica, drones, internet de las cosas y realidad virtual [43].

NXP

NXP -uno de los socios de Hardware de Google- provee plataformas de Hardware (Pico, Aragon, SprIot) de alto rendimiento y bajo consumo de energía que está optimizado para aplicaciones de Internet de las Cosas [44].

Phidgets

Un Phidgets es una representación física o implementación de un GUI widget⁹ y está integrado principalmente por sistemas de componentes electrónicos de bajo costo y sensores que están controlados por una computadora personal. Phidgets utiliza USB como interfaz de comunicación. La complejidad es administrada detrás de una API. Se pueden desarrollar aplicaciones en Mac OS, Linux, Windows CE y Sistemas Operativos Windows [45].

BeagleBone

Es una computadora pequeña del tamaño de una tarjeta de crédito, donde se puede ejecutar un sistema operativo, como ser Linux o Android. Puede definirse como un mini computador donde se pueden ejecutar programas sobre estos sistemas operativos [46].

Nanode

Nanode fue desarrollado en el Reino Unido y es una evolución de Arduino que permite conectarse a Internet a través de una API. Al igual que Arduino, se programa con el mismo entorno y es abierto. Nanode está disponible para que se pueda programar desde cualquier sistema operativo (MAC, Linux y Windows) [47].

Wasp mote

Libelium Wasp mote es un dispositivo diseñado para crear redes inalámbricas de sensores de muy bajo consumo y destinados a ser desplegados en un escenario real. El dispositivo comprende: un microcontrolador, memoria, batería, acelerómetro y enchufes para añadir módulos. La placa Wasp mote utiliza el mismo entorno de desarrollo de Arduino [48].

⁹ En informática un widget es una pequeña aplicación, usualmente representada de manera muy visual, forman parte de nuestros escritorios. Entre sus objetivos están mostrar y dar fácil acceso a algunas de las principales funciones del terminal.

Software

Sistemas Operativos

Contiki

Es un sistema operativo de código abierto para el IoT. Contiki conecta pequeños microcontroladores de bajo costo y bajo consumo a Internet. Proporciona un desarrollo rápido y fácil. Las aplicaciones se escriben en lenguaje C estándar. Es gratuito para sistemas comerciales y no comerciales con código fuente completo [25].

Mbed OS

Es un sistema operativo para dispositivos del Internet de las Cosas. Está especialmente diseñado para funcionar en entornos energéticos de bajo costo. El sistema operativo proporciona funcionalidades de conectividad, seguridad y administración de dispositivos requeridas por los dispositivos del Internet de las Cosas. Proporciona el framework C++ para el desarrollo de aplicaciones y admite todos los estándares abiertos clave para la conectividad y la administración de dispositivos [25].

TinyOS

Es un sistema operativo basado en componentes de Software libre y de código abierto para el Internet de las Cosas. TinyOS está diseñado para dispositivos inalámbricos de baja potencia, como los utilizados en redes de sensores, redes de área personal, edificios inteligentes y medidores inteligentes. Está basado en componentes nesC (Network Embedded System C) los cuales se construyen a partir de componentes ensamblados y escritos (en el lenguaje de programación C) para formar programas completos [25].

Micrium OS

Es el sistema operativo en tiempo real para los dispositivos integrados utilizados en el IoT. Soporta una amplia gama de dispositivos de microcontrolador y está diseñado para dispositivos de bajo consumo energético [25].

RIOT

Es un sistema operativo para los dispositivos del IoT, es el más popular y confiable y se basa en un microkernel. RIOT está diseñado para la eficiencia energética y soporta el desarrollo independiente del Hardware. RIOT es el único sistema operativo que proporciona un alto grado de modularidad para los dispositivos IoT. Este sistema operativo también ha incorporado soporte multiproceso que gestiona los recursos de los dispositivos de manera más eficiente. RIOT proporciona el framework de C/C++ para el desarrollo de aplicaciones [25].

Brillo

Es el sistema operativo totalmente nuevo para IoT introducido por Google en mayo de 2015. Brillo se basa en los niveles más bajos de Android que pueden ejecutarse con los requisitos mínimos del sistema. Es perfecto para dispositivos como: bombillas, tomacorrientes, interruptores, etc. [25].

Plataforma IoT

Xively

Es una plataforma IoT empresarial y una solución de aplicación. Xively antes conocido como Pachube permite a los usuarios conectar sus dispositivos a IoT y ayudar a administrar los objetos conectados. Asimismo ofrece conectividad segura, escalable, confiable y también ayuda en la construcción de aplicaciones empresariales y servicios de procesamiento de datos. Xively provee libertad para la conectividad, descubrimiento, administración de dispositivos, SDK para el desarrollo de aplicaciones y servicios en la nube para la gestión de datos [25].

Kaa

Es una plataforma de middleware de Código Abierto para IoT. Kaa es una plataforma confiable y segura para desarrollar objetos conectados. Kaa es de Código Abierto y se puede desplegar en cualquier lugar incluyendo la nube. Además soporta una gran variedad de dispositivos de Hardware y ofrece una comunicación segura para dispositivos a través del cifrado AES. El SDK de Kaa está disponible para C, C ++ y Java. El servidor de Kaa provee servicios de gestión de datos, integración,

visualización y muchos otros relacionados con los datos. La plataforma Kaa soporta también muchos servicios de datos como mongoDB, Hadoop, Spark, Oracle DB, etc. [25].

IBM Bluemix

Es un entorno de plataforma como servicio desarrollado por IBM. Soporta varios lenguajes de programación y servicios de forma integrada para crear, ejecutar, desplegar y gestionar aplicaciones en la nube. Bluemix es capaz de soportar: Java, Node.js, GO, PHP, Python, Ruby on Rails, etc. Puede definirse también como un conjunto de aplicaciones centrado en el entorno de ejecución, los contenedores de IBM y máquinas virtuales que permite al usuario crear servicios para el sistema IoT. En definitiva, ayuda a los usuarios a integrar las aplicaciones y ejecutar sistemas a través de un entorno seguro [25].

Carriots

Es una plataforma que está diseñado para IoT. Carriots permite: recopilar datos de varios dispositivos y almacenarlos, construir potentes aplicaciones utilizando el motor de SDK además de desplegar y escalar servicios a una variedad de dispositivos. Por otro lado también proporciona potentes API y servicios web para integrar aplicaciones. Con Carriots se puede construir servicios simples, confiables, escalables, rápidos, seguros y de bajo costo para el sistema IoT. Proporciona un entorno de administración y panel de control personalizado donde el usuario puede administrar todas las entidades de Carriots [25].

Nimbits

Es una plataforma abierta que permite almacenar datos de sensores en la nube. Nimbits se compone de: Nimbits Server, Nimbits.io, Nimbits Android y Nimbits Cloud. Nimbits Server permite a los usuarios grabar, almacenar y procesar datos de sensores. Nimbits.io es una biblioteca de código abierto de Java que permite a los usuarios crear Software para IoT. Nimbits Android es una aplicación desarrollada usando Por su parte, Nimbits.io y disponible en Google Play que ofrece muchas características de un cliente Nimbits. Nimbits Cloud es la instancia del servidor Nimbits. Por último, Nimbits Server se puede instalar en un microcontrolador, en un servidor y en la nube, además permite al usuario controlar, administrar y monitorizar dispositivos IoT desde la nube [25].

Capítulo 3

Internet de las Cosas y tecnologías de Google

En esta sección se explicarán las tecnologías necesarias para implementar productos del IoT con tecnologías de Google. Para ello Google propone tres tecnologías: Android Things, Weave y Google Cloud Platform [9].

3.1 Android Things

Es una nueva versión de Android destinada a permitir un desarrollo más rápido de los dispositivos conectados para el Internet de las Cosas. Android Things es un sistema operativo ligero que incluye Weave una plataforma de comunicación que permite la comunicación local y en la nube a través de múltiples protocolos inalámbricos. Por otra parte también ofrece seguridad a la escala que se precise a través de actualizaciones que administra Google y arranque verificado [49].

3.2 Weave

Google Weave, es la plataforma de comunicación de objetos del Internet de las Cosas que permite conectar objetos entre sí con dispositivos como teléfonos móviles y con la nube. Se podría decir que es un protocolo común basado tanto en la cercanía (Bluetooth), y en las redes (Wifi) como en la nube, permitiendo que diferentes objetos del IoT puedan interactuar entre ellos; como por ejemplo: que la lavadora se active cuando la puerta se cierre o controlar éstos desde un teléfono móvil estando fuera de casa a través de la nube [50].

3.3 Google Cloud Platform

Google Cloud Platform (a partir de ahora GCP) es una plataforma de computación en la nube, utilizado para crear ciertos tipos de soluciones a través de tecnologías almacenadas en la nube y permite destacar la rapidez y escalabilidad de su infraestructura. GCP se refiere un espacio virtual a través del cual se puede realizar

una serie de tareas que antes requería Hardware o Software. En lugar de esto se utiliza la nube de Google como único lugar de acceso, almacenamiento y gestión de datos [51].

3.4 Arquitectura

La propuesta de Google para el Internet de las Cosas es el de cubrir aspectos importantes del mismo como: computación en dispositivos, comunicación entre dispositivos y la gestión de datos generados por los dispositivos. Para llevar acabo tal propuesta Google plantea dos arquitectura del IoT: una basado en Weave y el otro basado en Google Cloud Platform [52] [50].

3.4.1 IoT basado en Weave

La propuesta de Google para el IoT basado en Weave es relativamente nuevo y fue implementado después de la compra de la empresa Nest Labs el año 2014 [53]. Weave está pensado para la conexión rápida de dispositivos electrónicos del hogar con Internet (bombillas, tomacorrientes, televisores, interruptores, etc.) para las marcas: Nest, Philips Hue, Samsung SmartThings, entre otros. Weave no se encuentra limitado a los dispositivos de las marcas antes mencionadas, sino que permite la integración de cualquier otro dispositivo electrónico que implemente el protocolo de comunicación Weave, por ejemplo, dispositivos desarrollados con Android Things.

El objetivo de Google con la presente propuesta, aparte de facilitar la integración de dispositivos del hogar, es la de cubrir necesidades de conexión entre dispositivos concediendo un protocolo de comunicación llamado Weave, un SDK, un Servidor en la nube, una Consola de Desarrollador y una aplicación de teléfono móvil para el control de los dispositivos IoT.

En la Figura 4, se observa la comunicación entre dispositivos Android Things, la nube de Weave y teléfonos móviles.

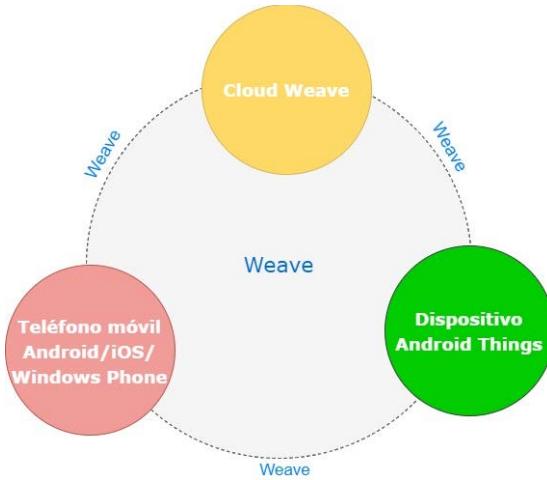


Figura 4: Integración Android Things - Cloud Weave.

Por medio del dispositivo Android Things se pretende dotar la capacidad de computación y comunicación a los pequeños dispositivos dispersos por el mundo. Estos dispositivos estarán conectados con sensores y actuadores los cuales permitirán recopilar datos de su contexto y tendrán la función de procesar los datos recopilados de manera local o enviarlos a la nube de Weave para su procesamiento.

Cloud Weave pretende dotar de una plataforma de computación en la nube escalable que permita recopilar y almacenar información de dispositivos implementados con el protocolo Weave o dispositivos Android Things.

Con los teléfonos móviles se pretende controlar el estado de los dispositivos por medio de una aplicación móvil ofrecido por Weave. La aplicación móvil se encuentra disponible para los sistemas operativos Android, iOS y Windows Phone.

3.4.2 IoT basado Google Cloud Platform

La otra propuesta de Google para el IoT es la basada en Google Cloud Platform para cubrir aspectos importantes del mismo como: computación, comunicación, procesamiento, análisis y la gestión de datos generados por los dispositivos IoT.

Como se aprecia en la arquitectura de la Figura 4.1, es similar a la arquitectura IoT basado en Weave, con la diferencia que la comunicación se realiza a través de una API de comunicación facilitado por GCP. También se observa la inexistencia de

comunicación directa entre los dispositivos Android Things y los Teléfonos móviles, puesto que toda comunicación entre ambos se lo realiza por medio de la nube GCP.

Los componentes que integran la arquitectura basado en GCP son: los dispositivos Android Things, los Teléfonos Móviles Android y el servicio de computación en la nube GCP.

Con Android Things se pretende dotar de la capacidad de computación y comunicación a cualquier objeto IoT. Asimismo, la comunicación entre los componentes y la gestión de datos se efectuará mediante GCP.

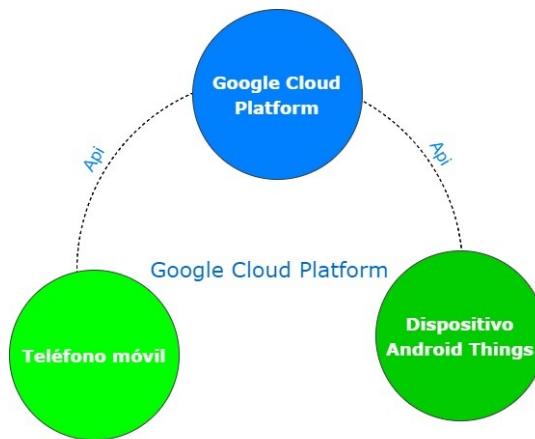


Figura 4.1: Integración Android Things - Google Cloud Platform.

3.5 Fabricantes y clientes asociados a Google

Entre los fabricantes de dispositivos y clientes asociados [51] que promueven la implementación Android Things, Weave y Google Cloud Platform encontramos los siguientes que detallamos en la lista de la Tabla 2.

Tabla 2: Fabricantes y clientes de Google Cloud Platform.

Android Things	Weave	Google Cloud Platform
Intel	Philips Hue	Netflix
NXP	Samsung Smartthings,	Spotify
Microchip	TP-Link	Coca Cola
Sierra Wireless	Belkin	Snapchat
ARM	WeMo	HTC
Tellmeplus	LiFX	Motorola
AllWiner	Honeywell	Philips
Mnubo	Wink	Niantic
Sotec	First Alert	Evernote
Realteck		Zulily
Losant		AirBus
Mongoose os		PocketGems
Actions		Fis
Helium		Khan Academy
Marwell		Ocado
Raspberry Pi		Heathrow
Adafruit		Atomic Fiction
Pimoroni		Best Buy
Sparkfun		
DigiKey		

Capítulo 4

Android Things

4.1 ¿Qué es Android Things?

Android Things es una nueva versión de Android destinada a permitir un desarrollo más rápido de los dispositivos conectados para el Internet de las Cosas. Se trata de un sistema operativo ligero que incluye Weave, una plataforma de comunicación, que permite la comunicación local y con la nube a través de múltiples protocolos inalámbricos. Android Things ofrece seguridad a escala que se precise a través de actualizaciones que administra Google y arranque verificado [49]. En la Figura 5 se observan objetos funcionando con Android Things.



Figura 5: Objetos funcionando con Android Things [54].

4.2 Evolución

En Google I/O 2015 -congreso de desarrolladores web organizado anualmente por Google-, se presentó el sistema operativo Brillo, basado en Android, creado directamente para el Internet de las Cosas; no se trataba de un sistema móvil propiamente dicho, sino de un Software derivado que permitía aprovechar una conectividad total, flexibilidad para adaptarse a cualquier cosa [55].

En diciembre de 2016, Google lanzó una versión de Android llamado Android

Things, y para ello lo que realizó fue unir el sistema operativo Brillo con algunos servicios de la plataforma de desarrolladores de Android (Android Studio¹⁰, SDK de Android¹¹ y Servicios de Google¹²) para facilitar el proceso de creación de productos de Internet de las cosas. De ahí nace Android Things una plataforma pensada para dispositivos que necesitan un sistema operativo para estar conectados y realizar tareas inteligentes [9].

4.3 Requisitos de Hardware y Software

4.3.1 Hardware

Android Things por el momento es compatible solo con las siguientes plataformas de Hardware detalladas en la Tabla 3.

Tabla 3: Plataforma de Hardware compatibles con Android Things [56].

Intel Edison	Intel Joule	NXP Pico	Raspberry Pi 3
			

Intel Edision

Intel Edison es un módulo de cómputo pequeño y potente en un chip que incluye un CPU, memoria, almacenamiento y doble banda Wifi y Bluetooth. El módulo se puede montar en una placa de expansión lo que permite la creación rápida de prototipos y aplicaciones del IoT.

¹⁰ Android Studio, es un entorno de desarrollo integrado oficial para la plataforma Android.

¹¹ SDK de Android, incluye un conjunto de herramientas de desarrollo. Comprende un depurador de código, biblioteca, un simulador de teléfono, documentación, ejemplos de código y tutoriales.

¹² Servicios de Google, es una aplicación de Android que permite que las aplicaciones del dispositivo estén siempre actualizadas.

Intel Joule

Intel Joule es una placa computacional o micro computador, y su módulo es un chip de bajo consumo, pensado para utilizar en el desarrollo de visión computacional, robótica, drones, internet de las cosas y realidad virtual.

NXP Pico

NXP Pico es uno de los socios de Hardware de Google, es una plataforma de Hardware de alto rendimiento y bajo consumo de energía esta optimizado para aplicaciones de IoT.

Raspberry Pi 3

Raspberry Pi 3 modelo B, es la tercera generación de Raspberry Pi, es un computador de placa reducida de bajo coste.

Las especificaciones técnicas de las plataformas de Hardware compatible con Android Things se lo detalla en la Tabla 4.

Tabla 4: Especificación técnicas de las plataformas de Hardware.

Plataforma	NXP				Raspberry PI	Intel	
	Pico i.MX7D	Pico i.MX6UL	Argon i.MX6UL	SprIoT i.MX6UL	Raspberry Pi 3	Edison	Joule
CPU y Memoria	-NXP i.MX7D -1GHz dual-core ARM Cortex A7 -512MB RA	-NXP i.MX6Ultralite -500MHz ARM Cortex A7 -512MB RAM	-NXP i.MX6Ultralite -500MHz ARM Cortex A7 -512MB RAM	-NXP i.MX6Ultralite -500MHz ARM Cortex A7 -512MB RAM	-Broadcom BCM2837 -1.2GHz quad-core ARM Cortex A53 -1GB RAM	-Intel® Atom™ -500MHz dual-core x86 -1GB RAM	-Intel® Atom™ -1.5GHz/1.7GHz quad-core x86 -3GB/4GB RAM
GPU	-N/A	-N/A	-N/A	-N/A	-VideoCore IV	-N/A	-Intel Gen9
Almacenamiento	-4GB eMMC	-4GB eMMC	-4GB eMMC	-4GB eMMC	-MicroSD card slot	-4GB eMMC	-8GB / 16GB eMMC
Monitor	-DSI	-No	-No	-No	-HDMI + DSI	-No	-HDMI
Cámaras	-CSI-2	-No	-No	-No	-CSI-2	-No	-CSI-2
Audio	-3.5mm Analógico	-3.5mm Analógico	-3.5mm Analógico	-3.5mm Analógico	-USB 2.0 -3.5mm Analógico	-USB 2.0	-USB 2.0
Redes	-10/100/1000 Ethernet -Wi-Fi 802.11ac (2.4/5.0GHz)	-10/100 Ethernet -Wi-Fi 802.11n (2.4GHz)	-10/100 Ethernet -Wi-Fi 802.11n (2.4GHz)	-10/100 Ethernet -Wi-Fi 802.11n (2.4GHz)	-10/100 Ethernet -Wi-Fi 802.11n (2.4GHz) -Bluetooth® 4.1	-Wi-Fi 802.11n (2.4/5.0GHz) -Bluetooth® 4.0	-Wi-Fi 802.11ac (2.4/5.0GHz) -Bluetooth® 4.2

	-Bluetooth® 4.1	-Bluetooth® 4.1	-Bluetooth® 4.1	-Bluetooth® 4.1			
Usb	-1x USB 2.0 Host -1x USB 2.0 OTG	-4x USB 2.0 Host	-1x USB 2.0 OTG	-2x USB 2.0 Host 1x USB 3.0 OTG			

4.3.2 Software

Kit de desarrollo de Software

El desarrollo de aplicaciones para Android Things es muy similar al desarrollo de aplicaciones para Android, pues implica escribir aplicaciones utilizando Java y XML en Android Studio o cualquier otro IDE compatible con el SDK de Android. Todo lo que se necesita es una plataforma de Hardware (Raspberry Pi 3, Intel Joule, Intel Edison o versiones compatibles de NXP) con el sistema operativo Android Things y los periféricos necesarios.

Versiones

Desde la publicación de Android Things en diciembre del 2016, hasta la actualidad se lanzaron siete versiones previas y según el sitio web de Android Things se tiene prevista tener nuevas versiones preliminares aproximadamente cada 6 a 8 semanas [57]. Al ser Android Things una versión previa, tiene varios problemas que se han ido corrigiendo. Estos problemas han sido detectados por la comunidad de desarrolladores y están relacionados con: la conectividad mediante bluetooth y wifi, accesos a las interfaces de entrada y salida de Hardware, problemas en tiempo de conexión al servicio de Google Play, problemas de compatibilidad entre plataformas de Hardware, entre otros. Estos problemas de funcionamiento se puntualizan a continuación en la Tabla 5.

Tabla 5: Problemas conocidos en las versiones previas de Android Things.

Versiónes Previas	Fecha de lanzamiento	Problemas conocidos	Nuevo
1	Diciembre del 2016	-Sistema de gestión de energía desactivado. -API de Bluetooth desactivado. -API USB desactivado.	

		<ul style="list-style-type: none"> -Los permisos solicitados por la aplicación no se conceden hasta el siguiente reinicio. -Google Play Services requiere de 2 a 3 minutos para arrancar. -El audio de grabación no es compatible con Intel Edison. -El soporte de la cámara para Intel Joule no está activada. -La red Wifi no puede conectarse a Internet. -La aceleración de gráficos de Hardware no está habilitado 	
2	Febrero 2017	<ul style="list-style-type: none"> -Continúan los mismos problemas conocidos de la versión anterior. 	<ul style="list-style-type: none"> -Soporte para la plataforma de Hardware Intel Joule. -Acceso a los periféricos de entrada/salida desde C y C++. -Habilitación del Kit de Desarrollo Nativo (NDK). -Soporte de Audio para las plataformas de Hardware Intel Edison, Intel Joule, Raspberry Pi. -Habilitación de la biblioteca de aprendizaje automático TensorFlow -Inspección del estado de puertos periféricos durante el desarrollo y la depuración
3	Abril 2017	<ul style="list-style-type: none"> -Continúan los mismos problemas conocidos de la versión anterior. -Problemas en la calidad de audio. 	<ul style="list-style-type: none"> -Android Things compatible con la plataforma de Hardware NXP Argon i.MX6UL. -Habilitación de la API para Bluetooth. -Habilitación de la API para USB. -Documentación de referencia en línea.
4	Mayo 2017	<ul style="list-style-type: none"> -Continúan los mismos problemas conocidos de la versión anterior. 	<ul style="list-style-type: none"> -Android Things compatible con la plataforma de Hardware NXP i.MX7D.

4.1	Junio 2017	-Continúan los mismos problemas conocidos de la versión anterior.	<ul style="list-style-type: none"> -Android Things compatible con la plataforma de Hardware NXP i.MX6UL Pico. -Nueva variante de Google Play Services para el Internet de las Cosas.
5	Agosto 2017	-Continúan los mismos problemas conocidos de la versión anterior.	<ul style="list-style-type: none"> -Adecuación de Android Things a Android O. -Android Things compatible con la plataforma de Hardware NXP SprIoT i.MX6UL Pico. -Google discontinua soporte para Intel Edison y Intel Joule. -Introducen servicio de aplicación para controlar el estado del dispositivo tales como restauración de fábrica o reinicio del dispositivo. -Las aplicaciones están obligados a solicitar permiso para administrar los Driver registrados. -Soporte para la biblioteca de gráficos OpenGL 2.0.
5.1	Agosto 2017	-Continúan los mismos problemas conocidos de la versión anterior.	<ul style="list-style-type: none"> -Detección automática de la configuración de Pantalla de Raspberry Pi. -Corrección del cursor asociados con un Mouse USB para ocultar/mostrar de forma incorrecta.

4.4 Arquitectura de Android Things

La arquitectura de Android Things consta de siete capas. En la Figura 6 se exponen las capas de la arquitectura de Android Things.

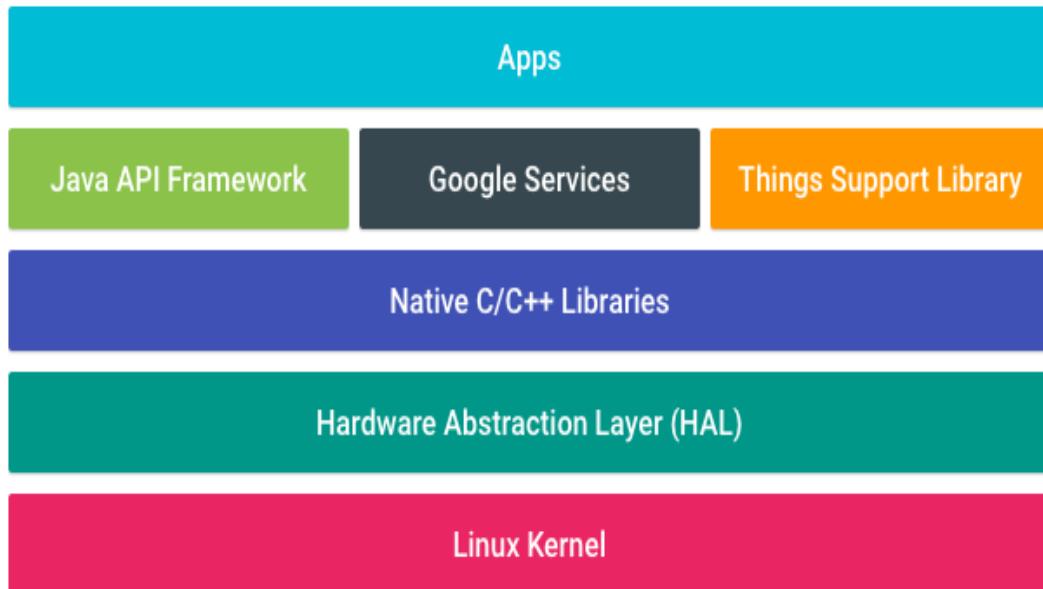


Figura 6: Arquitectura Android Things [56].

Núcleo de Linux

La base de la plataforma Android Things es el núcleo de Linux que contiene los controladores del Hardware. Este núcleo se encarga de manejar los recursos del Hardware de los dispositivos como: la CPU, la memoria, medidas de seguridad. Por ejemplo, el tiempo de ejecución de la máquina virtual de Android Things (ART – Android Runtime) se basa en el núcleo de Linux para funcionalidades subyacentes, como la generación de subprocesos y la administración de memoria de bajo nivel.

Capa de abstracción de Hardware

La capa de abstracción de Hardware (HAL – Hardware Abstraction Layer) brinda interfaces estándares a través de bibliotecas que exponen las capacidades de Hardware del dispositivo al framework de la API de Java. La HAL consiste en varios módulos de biblioteca y cada uno de estos implementa una interfaz para un tipo específico de

componente de Hardware, como el módulo de la cámara o de bluetooth. Cuando el framework de una API realiza una llamada para acceder a Hardware del dispositivo, el sistema Android Things carga el módulo de biblioteca para el componente de Hardware en cuestión.

Bibliotecas C/C++ nativas

Muchos componentes y servicios centrales del sistema Android Things, como la máquina virtual ART y la HAL, se basan en un código nativo que requiere bibliotecas nativas escritas en C y C++. La plataforma Android Things proporciona la API del framework de Java para exponer la funcionalidad de algunas de estas bibliotecas nativas a las apps. Por ejemplo, se puede acceder a OpenGL a través de la API de Java OpenGL del framework de Android para agregar a aplicaciones, compatibilidad con los dibujos y la manipulación de gráficos 2D y 3D.

Framework de la API de Java

El Framework de la API de Java son los cimientos para crear aplicaciones de Android Things escritas en el lenguaje de programación Java. Todo el conjunto de funciones del sistema operativo Android Things está disponible mediante API escritas en el lenguaje Java. Al igual que en Android, en esta capa se manejan los conceptos de: Activity Manager, View System, Resource Manager, etc. Con la diferencia que por ser Android Things una versión modificada de Android y pensado para dispositivos de bajo coste, no incluye las siguientes API: calendario, lista de contactos, administrador de documentos, gestor de descargas, ajustes, telefonía, diccionario de usuario y correo de voz.

Servicios de Google

Google Services es una aplicación del sistema de Android. El sistema operativo Android Things también incluye Google Services, lo que permite tener las aplicaciones de los dispositivos siempre actualizadas, ya que se encargará de comprobar que las aplicaciones instaladas estén en la última versión disponible. Las funciones principales de Google Services están relacionadas con: la autenticación del usuario, actualización de aplicaciones, los ajustes de privacidad, en la velocidad de las búsquedas y algunos servicios de localización.

Biblioteca de soporte de Cosas

Biblioteca de soporte de Cosas, es una nueva biblioteca desarrollada por Google que se encarga de la comunicación con los periféricos y controladores. Esta es una biblioteca completamente nueva que no está presente en el SDK de Android; además es una de las características más importantes de Android Things. La biblioteca expone un conjunto de Interfaces y Clases de Java que se pueden usar para conectar e intercambiar datos con dispositivos externos como sensores, actuadores, etc. Esta biblioteca oculta los detalles de comunicación interna, soportando varios protocolos estándar de la industria tales como: GPIO, I2C, PWM, SPI y UART.

Aplicaciones

El sistema operativo Android Things al ser una versión ligera de Android, por defecto no contiene ninguna aplicación preinstalada. La capa de aplicación tiene la función de administrar la aplicación desarrollada para el dispositivo Android Things. Todas las aplicaciones utilizan los servicios, las API y las bibliotecas de los niveles anteriores.

4.5 Características

4.5.1 Paquetes de aplicaciones

Android Things, al ser una versión modificada de la plataforma Android, no incluye las siguientes API en sus aplicaciones:

- Calendario (Información relacionada al calendario y eventos)
- Lista de contactos (Información relacionada con el contacto)
- Administrador de documentos (Permisos para acceder a los documentos lectura y escritura)
- Gestor de descarga (Gestión de descargas en segundo plano)
- Tienda de medios (Audio, video, archivo, imagen)
- Ajustes (Sonidos y vibración, pantalla, seguridad, batería, temas, etc.)
- Telefonía (SMS, MMS)
- Diccionario de Usuario (Palabras definidas para métodos de entradas, texto predictivo)

- Correo de voz (Registros de correos de voz reales)

4.5.2 Compatibilidad con los servicios de Google

Android Things es compatible con un subconjunto de las API de Google para Android. A continuación, en la Tabla 6 se desglosa el soporte de las API de Google para Android Things.

Tabla 6: API soportado y no soportado por Android Things.

API soportados	API no soportados
Cast	AdMob
Drive	Android Pay
Firebase Analytics	Firebase App Indexing
Firebase Cloud Messaging (FCM)	Firebase Authentication
Firebase Crash Reporting	Firebase Dynamic Links
Firebase Realtime Database	Firebase Invites
Firebase Remote Config	Firebase Notifications
Firebase Storage	Maps
Fit	Play Games
Instance ID	Search
Location	Sign-In
Nearby	
Places	
Mobile Vision	

Descripción de las API soportados

Google Cast: es un SDK que permite ampliar las capacidades de Android, Chrome o iOS para controlar un televisor o un sistema de sonido.

Google Drive: es un servicio para el alojamiento, sincronización y manipulación de archivos.

Firebase Analytics: es una solución analítica de aplicaciones que proporciona información estadística a los desarrolladores sobre la interacción de los usuarios con sus aplicaciones. La información se encuentra disponible como un tablero de mandos que proporciona información detallada de datos demográficos, artículos más comprados, etc.

Firebase Cloud Messaging: es una solución multiplataforma y gratuita que permite enviar, de forma gratuita y segura mensajes y notificaciones a una aplicación cliente.

Firebase Crash Reporting: es una solución que ayuda a diagnosticar y solucionar problemas creando informes detallados de una aplicación.

Firebase Realtime Database: es una base de datos alojada en la nube, los datos se sincronizan en tiempo real con cada cliente.

Firebase Remote Config: es un servicio en la nube que cambia la configuración y aspecto de aplicaciones sin publicar una nueva actualización.

Firebase Storage: es un servicio que ofrece la posibilidad de subir y descargar archivos de imagen, audio, video y otros de forma segura para aplicaciones de Firebase.

Fit: es un conjunto de API para facilitar la creación de aplicaciones de acondicionamiento físico.

Instance ID: es una API que proporciona un ID único por instancia de aplicaciones. El ID es único en todas las instancias de aplicaciones del mundo.

Location: es una API que permite localizar y ubicar un dispositivo por medio de un sencillo proceso técnico. Aprovechan los sensores (Global Positioning System - GPS) y señales de los dispositivos móviles, para dar a conocer las coordenadas geográficas de latitud y longitud. Permite crear potentes características basadas en el contexto con un impacto mínimo en los recursos del sistema. Combina y trabaja con siete señales: tiempo, ubicación, lugares, actividad, clima, auriculares y beacons (dispositivos pequeños basados en bluetooth de bajo consumo).

Nearby: es una API para conectar dispositivos cercanos, la función que tiene es traspasar información de un dispositivo a otro, para esta labor ambos dispositivos tienen que estar cercas y conectados a internet. Nearby utiliza para funcionar comunicación inalámbrica como Bluetooth, Bluetooth LE, Wifi y ultrasonidos. De esta manera podrán detectarse dispositivos y solicitar información. Se puede utilizar para enviar mensajes, conectar dispositivos cercanos, y trabajas con beacons.

Places: es una API que permite obtener la ubicación y puntos de interés que se actualizan regularmente mediante listas verificadas por el propietario y contribuciones moderadas por el usuario. Utiliza la misma base de datos que usan Google Maps y

Google+ para mostrar ubicaciones de lugares (Restaurantes, cines, parques, etc.).

Mobile Visión: es una API que proporciona un framework para encontrar objetos en fotos y videos. Incluye detectores que localizan y describen objetos visuales en las imágenes o fotogramas de video. Actualmente API Mobile Visión incluye detección de rostros, códigos de barra, detector de texto que se pueden aplicar por separado o conjunto.

4.5.3 Seguridad

Instance ID: es una API que proporciona un ID único por instancia de aplicaciones. Además de proporcionar identificaciones únicas para la autenticación, Instance ID puede generar tokens de seguridad para su uso con otros servicios. Asimismo, puede verificar la autenticidad de la aplicación, confirmar que el dispositivo de la aplicación está activo, identificar y realizar un seguimiento de las aplicaciones. Instance ID es única en todas las instancias de aplicaciones en todo mundo, por lo que su base de datos puede utilizarse para identificar de forma única y realizar el seguimiento de las instancias de aplicaciones.

Permisos a recursos: Android Things protege los recursos del Hardware e información personal con su sistema de permisos, esto es, una capa de seguridad en la que toda aplicación que quiera acceder a estos recursos tiene que declarar e informar al usuario a los recursos o información a acceder o modificar una determinada aplicación.

Los permisos no se solicitan en tiempo de ejecución, porque los dispositivos implícitos no están garantizados para tener una interfaz de usuario para aceptar el dialogo en tiempo de ejecución. Los permisos se deben declarar en el archivo “*manifest.xml*” de la aplicación. Todos los permisos declarados en el archivo “*manifest.xml*” se conceden en el momento de la instalación.

4.5.4 Pantallasopcionales

Android Things soporta las interfaces gráficas de usuario utilizando el mismo conjunto de herramientas de interfaz de usuario disponibles para aplicaciones Android con algunas diferencias:

- Android Things no incluye la barra de esta del sistema.

- Android Things no incluye botones de navegación.
- Android Things no requiere una pantalla, es opcional.

4.5.5 Notificaciones

Puesto que no hay barra de estado en Android Things, las notificaciones no se admiten.

4.5.6 Controladores de Usuarios

Los Controladores de Usuarios permiten a los desarrolladores de aplicaciones registrar nuevos controladores de dispositivo con el framework. Android Things introduce el concepto de controlador de usuario que consiste en componentes escritos desde aplicaciones que amplían los servicios del framework Android. Permite que cualquier aplicación inyecte eventos de Hardware en el framework para que otras aplicaciones puedan procesar utilizando el estándar de las API de Android. No se puede personalizar el comportamiento de los controladores de dispositivos en el núcleo de Linux, ni en la HAL para añadir nueva funcionalidad a un dispositivo. En la Figura 7 se muestra las capas gestionadas por Google y desarrolladores.

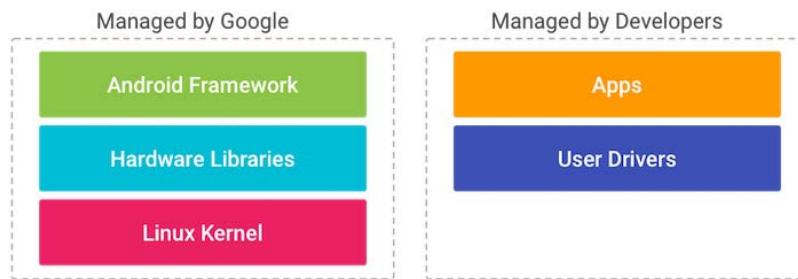


Figura 7: Capas gestionados por Google y desarrolladores [56].

Beneficios

Portabilidad: el código de la aplicación desarrollado para Android Things puede funcionar en una variedad de plataformas de Hardware y configuraciones sin abstracciones adicionales para la implementación del controlador de dispositivo.

Reusabilidad: puede extraer fragmentos de código y bibliotecas de Android existentes

en su aplicación sin la necesidad de modificarlos para manejar su implementación específica de Hardware.

Integración: Android Things a menudo combina datos de varios servicios juntos para mejorar la información reportada a aplicaciones. Los controladores de usuarios contribuyen a este proceso.

Tipos de controladores de dispositivos

GPS (Global Positioning System): GPS proporciona información de alta precisión sobre la ubicación física del dispositivo. Los módulos GPS son dispositivos de solo recepción que triangulan señales de los satélites remotos con el fin de determinar una ubicación física exacta. Los módulos GPS típicamente se conectan al sistema central a través de la interfaz UART, pero pueden utilizar otros periféricos de entrada y salida.

HID (Human Interface Devices): proporcionan las entradas de datos a las aplicaciones. Los teclados, touch pads, mouse, controles de juegos entre otros dispositivos que proporcionan este tipo de entrada. Los controladores de entrada permiten que los dispositivos interactuar con las API mejoradas de entrada del framework, tales como soporte de gestos y arrastrar y soltar.

Sensores: el framework de Android para sensores, soporta una amplia variedad de sensores para medir las condiciones del entorno físico y leer los datos en bruto de aplicaciones. Usando controladores de sensores, se pueden extender el framework de Android y añadir nuevos sensores conectados a través de periféricos de entrada/salida. Los datos de sensores son obtenidos a través de la API SensorManager de Android. El framework de Android para sensores, implementa la fusión de sensores para combinar los datos de varios sensores físicos en un solo sensor virtual.

4.5.7 Periféricos de Entrada y Salida

Android Things proporciona una API de periféricos de entrada y salida para comunicarse con sensores y actuadores utilizando protocolos e interfaces estándar de la industria. Los protocolos e interfaces estándar de la industria se clasifican en los siguientes:

- Entrada y salida de propósito general (GPIO – General Purpose

Input/Output).

- Modulación por ancho de pulso (PWM – Pulse Width Modulation).
 - Comunicación serial.
 - Circuito inter integrado (I2C – Inter Integrated Circuit).
 - Interfaz periférica serial (SPI – Serial Peripheral Interface).
 - Transmisor receptor asíncrono universal (UART-Universal Asynchronous Reciver Transmitter).

Entrada y salida de propósito general (GPIO – General Purpose Input Output)

GPIO es un pin genérico en una plataforma de Hardware, cuyo comportamiento se puede controlar en tiempo de ejecución. Los pines de GPIO, proporcionan una interfaz programable para leer el estado binario de un dispositivo de entrada o el control del estado binario de encendido y apagado de un dispositivo de salida. Los pines GPIO se pueden configurar como una entrada o salida o con un estado de alta o baja. Como una entrada la fuente externa determina el estado, y una aplicación puede leer el valor actual o reaccionar a los cambios de estado. Como una salida la aplicación es la que configura el estado del pin. En la Figura 8 se observan los pines GPIO.

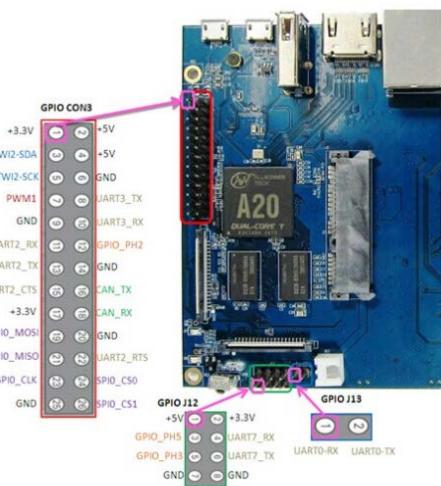


Figura 8: Pines de entrada y salida de propósito general [58].

Esta API se utiliza para sensores simples, tales como detectores de movimiento, detectores de proximidad, interruptores de nivel, botón pulsador que informa su estado como un valor binario de alta o baja. En la Tabla 7 se muestran algunos sensores para GPIO.

Tabla 7: Sensores para GPIO [58].

Sensor de movimiento	Sensor de proximidad	Botón pulsador
		

Modulación por ancho de pulso (PWM – Pulse Width Modulation)

La PWM de una señal, es una técnica para modificar el ancho de pulso, permite variar el voltaje de salida de un pin pudiendo pasar por valores intermedios. En la Figura 9 se describe un diagrama de serie de pulso entre 0%, 25% y 100%.

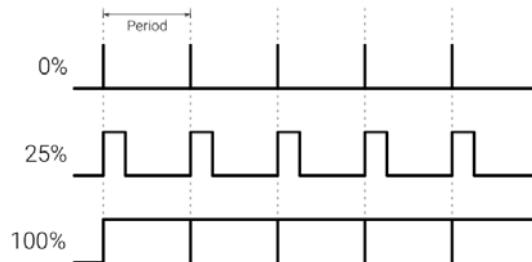


Figura 9: Diagrama de serie de pulso entre el 0%, 25% y 100% [56].

Las aplicaciones típicas para este tipo de señal son: controlar intensidad de un LED, mover servomotores, controlar LED RGB, controlar la velocidad de motores de corriente continua y controlar motores eléctricos o algún otro dispositivo que requiera una señal proporcional a un control preciso de la salida. En la Tabla 8 se muestran algunos dispositivos con soporte PWM.

Tabla 8: Dispositivos con soporte PWM [58].

Servomotor	Motor de corriente continua	Diodos led
		

Comunicación Serial

La comunicación serial es un protocolo muy común para la comunicación entre dispositivos que se incluye de manera estándar en cualquier computadora. La comunicación serial permite transferir grandes cantidades de datos entre dos o más dispositivos conectados en el mismo bus local. Los datos son enviados en función del tiempo, un bit detrás de otro. La comunicación serial en Android Things soporta los protocolos de la Tabla 9.

Tabla 9: Protocolos de serie.

Protocolo	Tipo de transferencia	Nro. de cables	Nro. De periféricos	Velocidad de transferencia
I2C	Sincrónico	2	Hasta 127	Bajo
SPI	Sincrónico	4+	Ilimitado	Alto
UART	Asincrónico	2 o 4	1	Medio

Circuito Inter Integrado (I2C – Inter Integrated Circuit)

I2C es un protocolo serie síncrono. I2C es muy utilizado para comunicar circuitos integrados, su uso más común es la comunicación entre un microcontrolador y sensores. La velocidad de transferencia es baja oscila entre 100 kb/s, 400 kb/s y 1 Mb/s. El I2C es un bus multi maestro es decir permite que haya múltiples maestros y múltiples esclavos en el mismo bus.

El bus I2C define dos pines, la de datos SDA (Serial Data) y la de reloj SCL (Serial Clock). La conexión I2C se describe en la Figura 10.

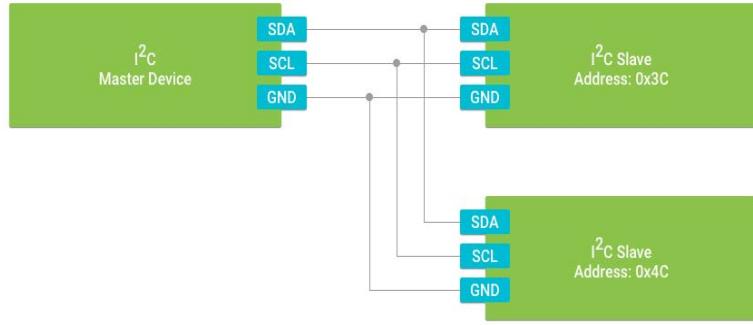


Figura 10: Conexión I²C [56].

Los sensores y actuadores son casos de uso común del bus I²C. El uso de este bus de datos incluye pantallas LCD, acelerómetros, sensores de temperatura, controladores de motor o cualquier circuito integrado que necesiten velocidades de transferencia de datos baja. Algunos módulos que soportan el protocolo I²C se muestran en la Tabla 10.

Tabla 10: Módulos con soporte I²C [58].

Pantallas LCD	Acelerómetros
Sensor de temperatura	Pines I ² C

Interfaz periférica serial (SPI – Serial Peripheral Interface)

SPI es un protocolo de comunicación serial síncrono que permite la comunicación a nivel de circuitos integrados. La transmisión de datos se realiza en serie, es decir un bit después de otro. El bus SPI posee una velocidad de transferencia de datos alta. La velocidad de transferencia se puede ajustar desde muy baja hasta muy alta a 10 Mb/s.

El bus SPI se define mediante cuatro pines:

CLK (Shared Clock Signal): señal de reloj compartida de bus. Esta señal rige la velocidad a la que se transmite cada bit.

MISO (Master Input Slave Output): Es la señal de entrada a un dispositivo, por aquí se reciben los datos desde el otro integrado.

MOSI (Master Output Slave Input): Transmisión de datos hacia el otro circuito integrado.

SS o CS: Chip Select o Slave Select, habilita el circuito integrado hacia el que se envían los datos, esta señal es proporcional en algunos casos no se usa. La conexión SPI se describe en la Figura 11.

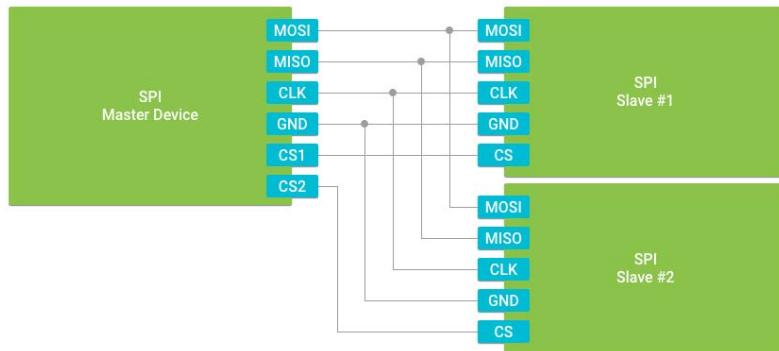


Figura 11: Conexión SPI [56].

El uso de este bus de datos incluye pantallas OLED, módulos Ethernet, Modulo tarjeta SD o cualquier circuito integrado que necesite velocidades de transferencia de datos alta. Algunos módulos que soportan el protocolo SPI se muestran en la Tabla 11.

Tabla 11: Módulos con soporte SPI [58].

Pantallas OLED	Modulo Ethernet



Transmisor Receptor Asíncrono Universal (UART – Universal Asynchronous Reciver Transmitter)

UART es un protocolo serie asíncrono. Es uno de los protocolos serie más utilizados. La mayoría de los microcontroladores disponen de Hardware UART. Controla los puertos y dispositivos serie, se encuentra integrado en la placa base o la tarjeta adaptadora del dispositivo. Es universal, por tanto, la velocidad de transferencia y formato de bytes de datos son configurables; y es asíncrono, por lo que no hay señales de reloj para sincronizar la transferencia de datos en los dispositivos.

Los periféricos UART normalmente son de dos tipos:

Puertos de 3 hilos incluyen, recepción de datos (RX), transmisión de datos (TX) y señales de referencia de tierra (GND).

Puertos de 5 hilos añaden la solicitud de envío (RTS) y limpieza de señales (CTS) usados para el control del flujo por Hardware. La conexión UART se describe en la Figura 12.

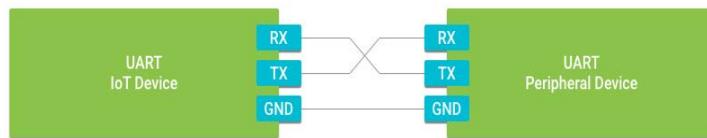
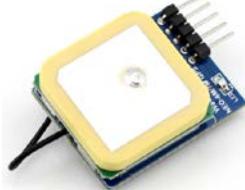
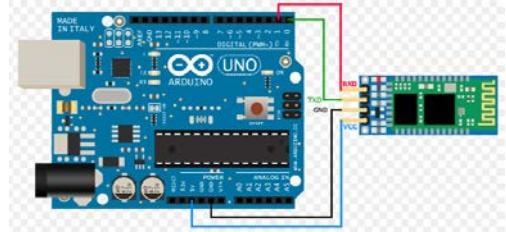


Figura 12: Conexión UART [56].

Dispositivos periféricos complejos, tales como módulos de GPS, pantallas LCD y XBee suelen utilizar el protocolo UART para comunicarse. Algunos módulos que soportan el protocolo UART se muestran en la Tabla 12.

Tabla 12: Módulos con soporte UART [58].

Modulo GPS	Módulo XBee
	
Modulo Wifi	Pines UART
	

API de periféricos nativos de entrada y salida (PIO Native – Peripheral Input/Output Native)

El API nativo de periféricos, permiten escribir código en C/C++ para el control de periféricos de entrada y salida de protocolos e interfaces estándares de la industria:

- GPIO (Entrada y Salida de Propósito General)
- PWM (Modulación por ancho de pulso)
- I2C (Circuito Inter Integrado)
- SPI (Interfaz Periférica Serial)
- UART (Transmisor Receptor Asíncrono Universal)

Esto permite escribir una aplicación de Android Things puramente en C/C++ o extender una aplicación de Android Things basados en Java con código C/C++ que utiliza el API nativo de periféricos.

4.6 Implementación de una aplicación para Android Things

Como parte de la implementación de un ejemplo básico en Android Things, en la presente sección se realizará una aplicación que encienda y apague un led

indefinidamente. Para su implementación se diseñará el prototipo de la Figura 13, se instalará Android Things sobre una Raspberry Pi y finalmente se creará una aplicación Java en Android Studio.

Requisitos

- Android Studio.
- Raspberry Pi con Android Things.
- Diodos Leds.
- Resistencia de 222 Ohmios.
- Cables de puente.
- Un protoboard.

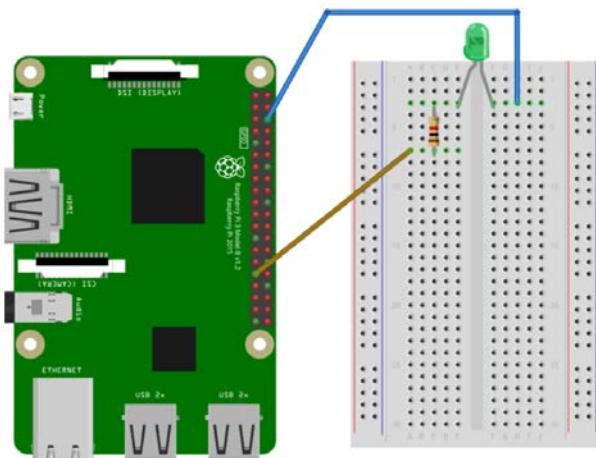


Figura 13: Diseño del prototipo [58].

Implementación de la Aplicación

La implementación de una aplicación para Android Things, se lo realiza de manera similar al desarrollo de aplicaciones para Android. Las aplicaciones de Android Things se implementa utilizando el lenguaje de programación Java, para ello es recomendable utilizar el IDE de desarrollo Android Studio ya que este integra una variedad de herramientas para el desarrollo de aplicaciones Android Things.

Por tanto, para el desarrollo de una aplicación Android Things es necesario realizar algunos ajustes en los siguientes archivos:

En el fragmento de Código 1 perteneciente al archivo de configuración

AndroidManifest.xml es necesario agregar las líneas 4 y 5 cuando la aplicación necesite conectarse al Internet. Las líneas 6 y 7 son permisos para habilitar accesos a los controladores de otros fabricantes.

```
1. <manifest xmlns:android="http://schemas.android.com/apk/res/android"
2.   package="com.example.androidthings.simplepio">
3.
4.   <uses-permission android:name="android.permission.INTERNET" />
5.   <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
6.   <uses-permission
7.     android:name="com.google.android.things.permission.MANAGE_INPUT_DRIVERS" />
8.   <uses-permission
9.     android:name="com.google.android.things.permission.MANAGE_SENSOR_DRIVERS" />
10.    <application android:allowBackup="true" android:icon="@android:drawable/sym_def_app_icon"
11.      android:label="@string/app_name">
12.        <uses-library android:name="com.google.android.things"/>
13.        <activity android:name=".BlinkActivity">
14.          <intent-filter>
15.            <action android:name="android.intent.action.MAIN" />
16.            <category android:name="android.intent.category.LAUNCHER" />
17.          </intent-filter>
18.          <!-- Launch activity automatically on boot -->
19.          <intent-filter>
20.            <action android:name="android.intent.action.MAIN"/>
21.            <category android:name="android.intent.category.IOT_LAUNCHER"/>
22.            <category android:name="android.intent.category.DEFAULT"/>
23.          </intent-filter>
24.        </activity>
25.      </application>
26.
27. </manifest>
```

Código 1: AndroidManifest.xml.

En el fragmento de Código 2, es obligatorio agregar la versión del SDK ya que las aplicaciones de Android Things se implementan con la versión 24 de Android hacia adelante, tal como se observa en la línea 4. También es necesario agregar la dependencia de la línea 23 para acceder a la API de Android Things.

```
1. apply plugin: 'com.android.application'
2.
3. android {
4.   compileSdkVersion 24
5.   buildToolsVersion '25.0.0'
6.
7.   defaultConfig {
8.     applicationId "com.example.androidthings.simplepio"
```

```

9.        minSdkVersion 24
10.       targetSdkVersion 24
11.       versionCode 1
12.       versionName "1.0"
13.    }
14.    buildTypes {
15.        release {
16.            minifyEnabled false
17.            proguardFiles getDefaultProguardFile('proguard-
   android.txt'), 'proguard-rules.pro'
18.        }
19.    }
20. }
21.
22. dependencies {
23.     provided 'com.google.android.things:androidthinks:0.2-devpreview'
24. }

```

Código 2: Build.gradle.

En el fragmento de Código 3, BlinkActivity es donde se escribe la lógica de la aplicación. Para acceder a los periféricos de entrada y salida de Raspberry Pi hay que importar las bibliotecas de las líneas 2 y 3. En la línea 12 se ha definido el intervalo para encender/apagar el led. En la línea 16 se ha instanciado un objeto de la clase Handler para la ejecución de tareas en segundo plano. Finalmente, en la línea 18 se ha definido un variable para acceder a los estados del Led.

```

1. import android.app.Activity;
2. import com.google.android.things.pio.Gpio;
3. import com.google.android.things.pio.PeripheralManagerService;
4. import android.os.Bundle;
5. import android.os.Handler;
6. import android.util.Log;
7.
8. import java.io.IOException;
9.
10. public class BlinkActivity extends Activity {
11.     private static final String TAG = "Blink";
12.     private static final int INTERVAL_BETWEEN_BLINKS_MS = 5000;
13.     private Handler mHandler = new Handler();
14.     private Gpio mLedGpio;

```

Código 3: BlinckActivity.java.

En el fragmento de Código 4, el método onCreate de la clase BlinckActivity, en la línea 4 se ha definido una instancia de la clase PeripheralManagerService; esta clase es la que implementa los métodos para el acceso a los periféricos del Hardware. En las líneas 6 a 9, se accede al pin donde se conectó el led, se establece el estado del led a

apagado y se inicia en segundo plano la ejecución de la aplicación.

```
1. @Override
2. protected void onCreate(Bundle savedInstanceState) {
3.     super.onCreate(savedInstanceState);
4.     PeripheralManagerService service = new PeripheralManagerService();
5.     try {
6.         String pinName = BoardDefaults.getGPIOForLED();
7.         mLedGpio = service.openGpio(pinName);
8.         mLedGpio.setDirection(Gpio.DIRECTION_OUT_INITIALLY_LOW);
9.         mHandler.post(mBlinkRunnable);
10.    } catch (IOException e) {
11.        Log.e(TAG, "Error on PeripheralIO API", e);
12.    }
13. }
```

Código 4: Método onCreate.

El método onDestroy del fragmento de Código 5, detiene la tarea ejecutada en segundo plano y cierra la conexión con el led.

```
1. @Override
2. protected void onDestroy() {
3.     super.onDestroy();
4.     mHandler.removeCallbacks(mBlinkRunnable);
5.     Log.i(TAG, "Closing LED GPIO pin");
6.     try {
7.         mLedGpio.close();
8.     } catch (IOException e) {
9.         Log.e(TAG, "Error on PeripheralIO API", e);
10.    } finally {
11.        mLedGpio = null;
12.    }
13. }
```

Código 5: Método onDestroy.

El objeto de la clase Runnable del fragmento de Código 6, es el encargado de realizar la ejecución de la tarea en segundo plano (infinitamente); esto para lograr que el Led se encienda y se apague cada cierto tiempo.

```
1. private Runnable mBlinkRunnable = new Runnable() {
2.     @Override
3.     public void run() {
4.         if (mLedGpio == null) {
5.             return;
```

```
6.         try {
7.             mLedGpio.setValue(!mLedGpio.getValue());
8.             mHandler.postDelayed(mBlinkRunnable, INTERVAL_BETWEEN_BLINKS_MS);
9.         } catch (IOException e) {
10.             Log.e(TAG, "Error on PeripheralIO API", e); }
11.     }
12. };
```

Código 6: Objeto de la clase Runnable.

Finalmente, en la Figura 14 se observa el montaje del circuito.

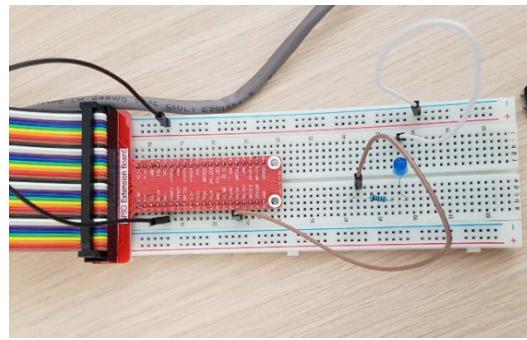


Figura 14: Montaje del circuito.

Capítulo 5

IoT basado en Weave

5.1 ¿Qué es Weave?

Google Weave es la plataforma de comunicación de objetos del IoT que permite conectar objetos entre sí, con dispositivos como teléfonos móviles y con la nube. Se podría decir que es un protocolo común basado tanto en la cercanía (Bluetooth), redes (Wifi) como en la nube permitiendo que diferentes objetos del IoT puedan interactuar entre ellos como por ejemplo que la lavadora se active cuando la puerta se cierre o controlar éstos desde un teléfono móvil estando fuera de casa a través de la nube [50]. En la Figura 15 se observan objetos conectados a través de Weave.



Figura 15: Objetos conectados a través de Weave [59].

5.2 Requisitos de Hardware y Software

5.2.1 Hardware

Por el momento Weave es compatible solo con los dispositivos de sus fabricantes asociados tales como termostatos, bombillas, tomacorrientes, televisores e interruptores de los fabricantes asociados. En caso que se necesite utilizar Weave en

el desarrollo de prototipos, este protocolo es soportado por Qualcomm QCA4010, Marwell MW302 o cualquier otro dispositivo que funcione sobre el sistema operativo Linux.

5.2.2 Software

- SDK Weave (Weave Device SDK)
- Servidor Weave (Weave Server)
- Consola de Desarrollador IoT (IoT Developer Console)
- Aplicación de Desarrollador Weave (Weave Developer App)
- Utilidades de Línea de Comandos (Command Line Tools)

SDK Weave

El SDK es un conjunto de herramientas ligero desarrollado en el lenguaje de programación C, se integra en los dispositivos de comunicación local y remota permitiendo conectar al Servidor Weave. El SDK es soportado por Linux, Qualcomm QCA4010 y Marwell MW302.

Servidor Weave

Este servicio en la nube maneja el registro del dispositivo, la transmisión de comandos, el almacenamiento de estados y la integración con los servicios de Google, como el asistente de Google.

Consola de Desarrollador IoT

La Consola de Desarrollador IoT es una aplicación web para desarrolladores de dispositivos. Proporciona una forma sencilla de administrar productos y cargar información de la interfaz de usuario acerca de dispositivos en los manifiestos del modelo. La consola IoT proporciona herramientas para acceder a dispositivos, realizar acciones de configuración y monitorización. La consola de desarrollo IoT se observa en la Figura 16.

Usando la consola IoT, se pueden:

- Configurar y registrar productos con el servidor Weave y proporcionar a los usuarios finales información sobre el dispositivo.
- Probar las funcionalidades de Weave controlando los dispositivos en desarrollo a través de la web.

- Monitorizar en tiempo real las activaciones y comandos usados en el dispositivo.

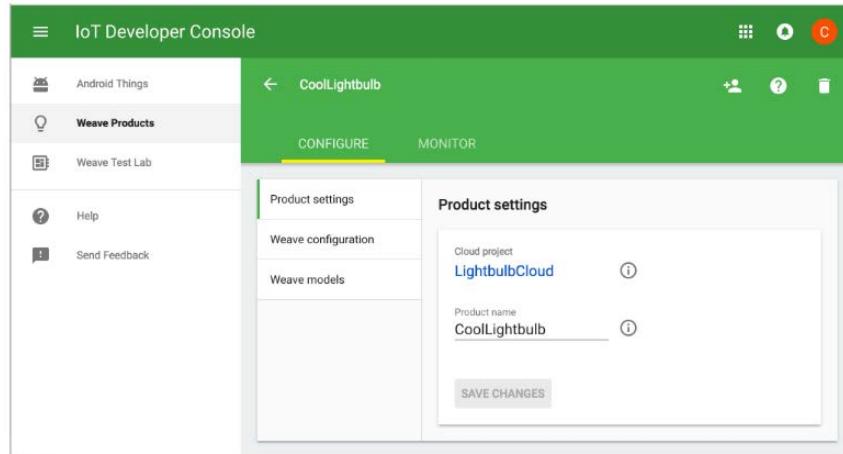


Figura 16: Consola de Desarrollador IoT [50].

Aplicación de Desarrollador Weave

La Aplicación de Desarrollador Weave es una aplicación para teléfonos móviles. Permite controlar todos los dispositivos Weave desde un teléfono móvil con Android o iOS. La aplicación, está destinada para la administración básica de dispositivos. Comprueba el registro de un dispositivo, administra el estado del dispositivo y transmite comandos hacia el dispositivo. La interfaz de la aplicación de Desarrollador Weave se observa en la Tabla 13.

Tabla 13: Aplicación del desarrollador Weave [50].



Utilidades de Línea de Comandos

Las utilidades de Línea de Comando son pequeños programas que se encuentran incluidos en el SDK para desarrolladores de Weave. Esta utilidad permite probar comandos, verificar estados, ver eventos e historia de estados de un determinado dispositivo Weave.

5.3 Seguridad de Weave

- Control de cuentas de usuario mediante cuenta de Google.
- Seguridad de la capa de transporte (TLS) + Encriptación de datos
- Todas las conexiones que envía con Weave hacia un dispositivo se encuentran encriptadas de principio a fin.

5.4 Dispositivo Weave

Un dispositivo Weave es un dispositivo IoT compatible con el protocolo Weave. Apoya el protocolo Weave proporcionando una descripción de las características del dispositivo utilizando componentes y funcionalidades. El conjunto de componentes y funcionalidades que describen un dispositivo específico se denomina esquema de dispositivo.

El protocolo Weave utiliza la información del esquema de dispositivo para determinar los comandos soportados por un dispositivo y qué tipo de información de estado puede mantener este.

5.5 Funcionalidad y componentes de dispositivo

Funcionalidad: Una funcionalidad define un atributo del dispositivo. Por ejemplo, una funcionalidad “onOff” describe un atributo para encender o apagar el dispositivo, también define el estado de encendido/apagado y los comandos para esa función.

Componente: Un componente describe la configuración o la relación lógica entre las funcionalidades que comprenden un dispositivo, como es estado actual de cada atributo. Por ejemplo, un interruptor puede proporcionar un componente “PowerSWITCH” para representar el interruptor. El componente “PowerSWITCH” utiliza la función “OnOff” para gestionar los comandos de encendido/apagado y el estado de actual del interruptor.

Un dispositivo Weave debe especificar una lista de componentes que describen el dispositivo y estado del dispositivo.

5.6 Información Obligatoria

Antes de crear un nuevo producto Weave se tiene que recopilarse la siguiente información:

- Cuenta de Google (Google Account)
- Proyecto Cloud Weave (Cloud Project)
- Código de Producto (Product Code Name)
- Tipo Dispositivo (Device Kind)
- Nombre del Modelo (Model Name)
- Descripción del Modelo (Model Description)

Cuenta de Google: tener una cuenta de Google asociada al producto. Después de crear un producto, puede agregar cuentas adicionales de Google a través de los permisos de Proyecto Cloud Weave.

Proyecto Cloud Weave: un proyecto en *Google Cloud Platform* ayuda a administrar credenciales de Weave y otros servicios de dispositivos. Cuando crea un producto Weave se tiene que asociar a un Proyecto Cloud Weave una vez asociado no puede cambiarlo posteriormente. Weave utiliza el Proyecto Cloud Weave principalmente para el control de acceso, permiso y para habilitar la API de Weave. Un producto Weave puede estar asociado con un solo Proyecto Cloud Weave, mientras que un Proyecto Cloud Weave puede asociar varios productos Weave.

Código de Producto: es el nombre interno para referirse a un producto Weave. Solo los desarrolladores Weave pueden ver el nombre del producto en la consola IoT.

Tipo de Dispositivo: es la categoría de un producto Weave. Los tipos pueden ser bombillas, tomacorrientes, interruptores, termostatos o televisores. El tipo determinara como se agruparán los dispositivos dentro de una Aplicación de Desarrollador Weave y los comandos, esquema de estado soportados por un dispositivo.

Nombre del Modelo: describe el nombre del modelo de dispositivo. Si un modelo tiene un nombre comercial, Weave recomienda utilizarlo seguido del identificador único del modelo entre paréntesis. Por ejemplo, XP-500(CGV150185). Si un modelo no tiene un nombre comercial, se debe establecer un identificador único de modelo. Por ejemplo, KFP0711OB.

Descripción del Modelo: es un breve párrafo que proporciona más detalles para identificar el producto, su función, y la información específica del modelo.

5.7 Esquema común de dispositivos

Un problema común de los dispositivos IoT es la falta de la interoperabilidad. Los dispositivos de fabricantes pueden tener diferentes funcionalidades o pueden representar la misma funcionalidad de manera diferente, lo que dificulta trabajar con otros dispositivos.

Weave resuelve el problema de interoperabilidad mediante la introducción de un conjunto de esquemas comunes y funcionalidades que representan tipos de dispositivos estándar, que los desarrolladores pueden usar para representar un dispositivo. Estos tipos de dispositivos hacen posible un estándar para que los clientes accedan a ellos.

Existen esquemas comunes para termostatos, bombillas, tomacorrientes, interruptores, televisores, etc. Un esquema común para tomacorrientes se describe en la Tabla 14.

Tabla 14: Esquema común para tomacorrientes [50].

Nombre del Componente	Característica	Observación
Power_switch	OnOff	Requerido
Regulador_intensidad	Brightness	Opcional

5.8 Guía de Dispositivos

Weave proporciona un conjunto de esquemas que representan los tipos de dispositivos comunes, tales como termostatos, bombillas, tomacorrientes, televisores e interruptores. Las guías describen los componentes del dispositivo para cada tipo de aparato. Si un dispositivo no es uno de los tipos de aparatos estándar incluidos en la guía, se puede elegir un esquema de similar funcionalidad y utilizarlo como una plantilla para la construcción de sus funcionalidades. Entre las guías de dispositivos Weave suministra:

- Guía para termostatos
- Guía para bombillas
- Guía para tomacorrientes
- Guía para televisores
- Guía para interruptores

Dependiendo de la Guía, estos incluyen funciones genéricas. Cada función tiene uno o varios estados, comandos y códigos de error.

Por ejemplo, según la documentación de Weave, la guía para termostatos (o esquema que representa el dispositivo termostato) estaría compuesta por las siguientes funciones genéricas y estados tal como se observa en la siguiente sección. Las guías restantes se pueden consultar en el Anexo A.

Guía para termostatos

Esta guía define un esquema de alto nivel para un dispositivo que controla la climatización, ventilación, aire acondicionado y el sistema de calefacción central de un hogar. Sus funciones genéricas y estados del termostato se describen en la Tabla 15.

Tabla 15: Guía para termostatos [50].

Termostato	
Funciones Genéricas	Estados
Device Describe la información del dispositivo como ser: fabricante, modelo.	<ul style="list-style-type: none"> Nombre del dispositivo. Ubicación. Descripción del dispositivo. Número de serie. ID generado por el dispositivo utilizado para la identificación en redes locales y el acceso sin conexión. Versión del firmware del dispositivo. Errores (valor fuera de rango, valor invalido).
HvacSubsystemController Describe un controlador genérico que puede tomar sus propias decisiones sobre cuando encender/apagar un subsistema.	<ul style="list-style-type: none"> Modo del controlador (deshabilitado, siempre encendido, automático). Estado del subsistema (encendido, apagado). Errores (valor inválido, estado inválido).
TempSetting Describe un ajuste de temperatura genérico que puede controlar una calefacción, aire acondicionado, refrigerador u otros dispositivos similares	<ul style="list-style-type: none"> Temperatura mínima. Temperatura máxima. Ajuste de la temperatura en grados centígrados. Manejo de Errores (Valor fuera de rango, estado invalido).
TempRangeSetting Describe una configuración genérica de la temperatura mínima y máxima para controlar una calefacción, un aire acondicionado, refrigerador u otros dispositivos similares.	<ul style="list-style-type: none"> Puntos de ajuste alto y bajo. Ajustes de los rangos alto y bajo de temperatura máxima. Ajustes de los rangos alto y bajo de temperatura mínima. Manejo de Errores (valor fuera de rango, estado inválido).
TempSensor Describe la temperatura adquirida desde un sensor.	<ul style="list-style-type: none"> Temperatura.
HumiditySensor Describe la humedad obtenida por el sensor de humedad en el dispositivo.	<ul style="list-style-type: none"> Humedad.

TempUnitsSetting Describe las unidades de medida seleccionados por el usuario en el termostato.	<ul style="list-style-type: none"> • Ajuste y obtención de unidades de temperatura (Celsius o Fahrenheit).
---	---

5.9 Desarrollo de dispositivo

Weave es un protocolo independiente del Hardware con un conjunto de requerimientos del sistema soportados por distribuciones de Linux, Qualcomm y Marvel SOC.

Para minimizar el esfuerzo de integración y garantizar implementaciones de dispositivos consistentes y seguros, Google provee un SDK también llamado “libiota” que es una biblioteca desarrollado en C y que implementa el protocolo Weave.

Integración y configuración de dispositivo

La integración de Weave está previsto para Linux, Marwell MW302 y Qualcomm QCA4010. Para el desarrollo con Weave, se necesitan establecer las opciones de configuración correctas para el producto y gestionadores de compilación para comandos y estados.

Comandos y estado

Cuando el SDK de Weave se está ejecutando en un dispositivo, en ese momento se construyen las definiciones de las funcionalidades y las descripciones de los componentes que necesitan los usuarios para interactuar con el producto final. Las definiciones de las funcionalidades autorizan a las aplicaciones clientes Weave, descubrir y usar la funcionalidad que provee el dispositivo; mientras que las descripciones de los componentes proporcionan información acerca de la relación lógica entre las funcionalidades del dispositivo y el estado actual de cada atributo.

Después de especificar las definiciones de funcionalidades y descripciones de componentes, se puede empezar a construir controladores de comandos para responder a los comandos entrantes que haya definido, y un estado de máquina para actualizar el estado del dispositivo. El SDK Weave gestiona la traducción de definiciones de funcionalidades y cambios en la descripción de componentes a las aplicaciones cliente y servicios web. El SDK también administra la vinculación de los controladores de comandos a los comandos entrantes.

Acceso

Para acceder a la API de Weave y a la Aplicación de Desarrollador Weave, debe registrarse en el grupo de Google Weave utilizando una cuenta de correo electrónico de Google. Al ser miembro de este grupo, se puede acceder a la API de Weave para conectar el SDK Weave al Servidor Weave y a la Aplicación de Desarrollador Weave, los cuales son utilizados para verificar integraciones de las tecnologías.

Se debe utilizar la misma cuenta de Google para acceder a la Consola de Desarrollador IoT y a la Aplicación de Desarrollador Weave.

Registro de dispositivo

El registro de dispositivos es el proceso de crear nuevas funcionalidades de dispositivos. Por su parte, el proceso de registro es el que se encarga de asociar un dispositivo en red con la cuenta de un usuario específico.

Desde una perspectiva de implementación, el proceso de registro incluye tres partes importantes:

- Descubrimiento.
- Canal seguro.
- Información de Registro.

Descubrimiento

El descubrimiento de dispositivos se produce en uno o más medios de transporte, dependiendo del Hardware disponible en el dispositivo. Esto depende del fabricante del dispositivo para proveer un mecanismo en la disposición de la conexión de red.

Canal seguro

Después de que el dispositivo es descubierto, el flujo de registro se mantiene seguro para intercambiar una pequeña cantidad de información.

Información de Registro

Cuando un canal seguro es establecido con el dispositivo, la aplicación registrada puede enviar información relevante hacia el dispositivo, incluyendo la información de registro en la nube y las credenciales de red.

Capítulo 6

IoT basado en Google Cloud Platform

6.1 ¿Qué es Google Cloud Platform?

Google Cloud Platform es una plataforma de computación en la nube y es utilizada para crear ciertos tipos de soluciones a través de tecnologías almacenadas en la nube, lo cual permite destacar la rapidez y escalabilidad de su infraestructura. Google Cloud Platform se refiere un espacio virtual a través del cual se puede realizar una serie de tareas que antes requerían Hardware o Software. En lugar de esto se utiliza la nube de Google como único lugar de acceso, almacenamiento y gestión de datos [52]. En la Figura 17 se observa los servicios de Google Cloud Platform agrupados en componentes.



Figura 17: Servicios¹³ de Google Cloud Platform agrupados en componentes [52].

¹³ En la Figura 17 no incluye el servicio “Cloud IoT Core” ya que el mismo fue introducido el mes de mayo del 2017 como versión “beta private” y recientemente el mes de octubre del 2017 fue habilitado para el público en general.

6.2 Requisitos de Hardware y Software

Google Cloud Platform al ser una plataforma de computación en la nube, no requiere de Software y Hardware especializado. El acceso a la plataforma se realiza a través de cualquier navegador con conexión a Internet.

6.3 Componentes de Google Cloud Platform

Google Cloud Platform ofrece una variedad de servicios basados en la nube. Se pueden dividir en seis grandes grupos:

- Procesamiento
- Almacenamiento y base de datos
- Redes
- Big Data
- Internet de las Cosas
- Aprendizaje Automático
- Identidad y Seguridad

6.3.1 Procesamiento

El componente de Procesamiento ofrece servicios para procesar datos, cálculo, renderizado de gráficos, análisis de datos o cualquier otra tarea específica que requiera mayor capacidad de procesamiento. De igual forma, oferta también servicios para la ejecución de máquinas virtuales, plataformas de *back-ends* móviles y aplicaciones web escalables. También permite ejecutar y administrar contenedores Docker.

Compute Engine

Servicio para la administración y ejecución de máquinas virtuales en la infraestructura de Google.

App Engine

Es una plataforma para crear *back-ends* móviles y aplicaciones web escalables. Proporciona servicios y API integrados, como almacenes de datos NoSQL, Memchache

y una API para la autenticación de usuarios. Escala aplicaciones de forma automática según la cantidad de tráfico que recibe.

Container Engine

Ejecuta contenedores en GCP. Es un poderoso administrador de clústeres y sistema de organización para ejecutar contenedores de Docker.

Cloud Function (beta)

Entorno sin servidores para compilar y conectar servicios en la nube.

6.3.2 Almacenamiento y base de datos

El componente de Almacenamiento y base de datos ofrece soluciones de almacenamiento en función de lo que se pretenda almacenar y la disponibilidad, la durabilidad y el rendimiento que deseemos para dicha información.

Cloud Storage

Permite el almacenamiento en todo el mundo y la recuperación de cualquier cantidad de datos en cualquier momento. Se puede utilizar para una serie de escenarios que incluyen servir contenido del sitio web, el almacenamiento de datos o la distribución de datos grandes a los usuarios a través de descarga directa.

Cloud SQL

Es un servicio de base de datos totalmente administrado para gestionar información de bases de datos relacionales tales como: Mysql o PostgreSQL.

Cloud Bigtable

Es un servicio de base de datos NoSQL de Google para trabajar con Big Data. Es la misma base de datos que se encuentra detrás de los servicios centrales de Google, como Búsqueda, Analytics, Mapas y Gmail.

Cloud Spanner

Es un servicio de base de datos relacionales y completamente administrado. Ofrece una semántica relacional tradicional (esquemas, transacciones, SQL). Las sintaxis para la gestión de las bases de datos corresponden al ANSI-2011 con extensiones.

Cloud Datastore

Es un servicio de base de datos NoSQL para datos no relacionales. Ofrece un gran número de funciones como las transacciones, consultas tipo SQL, índices o interfaz RESTful entre otros.

Persistent Disk

El disco persistente de Google es un almacenamiento de confianza y de alto rendimiento para instancias de máquinas virtuales. El disco persistente proporciona almacenamiento SSD y HDD que pueden conectarse a instancias que se ejecutan en Google Cloud Engine o Google Container Engine.

6.3.3 Redes

El componente de Redes ofrece soluciones para la gestión de las redes de una infraestructura en la nube. Para ello ponen a disposición herramientas que permiten crear subredes, firewalls, routers, VPN, NAT o hacer conexiones privadas a un centro de datos corporativo.

Virtual Private Cloud (VPC)

La red Virtual Private Cloud proporciona la funcionalidad de red gestionada para los recursos Google Cloud Platform. Mediante este servicio se puede gestionar las redes, router, cortafuegos direcciones IP Externas, rutas y VPN de los recursos de Cloud Platform.

Cloud Load Balancing

Servicio de balanceo de cargas para distribuir las peticiones (HTTP, TCP, UDP) de usuarios entre los conjuntos de instancias.

Cloud CDN

El Cloud CDN (Content Delivery Network) es un servicio que se puede utilizar para acelerar la entrega de contenido en caché de un sitio web o aplicación.

Cloud InterConnect

Servicio que permite conectar una infraestructura externa con la infraestructura de Google. Además, permite establecer una conexión dedicada a Cloud Platform, con lo que se evitarán recorridos innecesarios por redes de terceros.

Cloud DNS

Servicio de DNS. Cloud DNS traduce solicitudes de nombres de dominio en direcciones IP.

6.3.4 Big Data

El componente de Big Data ofrece soluciones como conseguir la máxima capacidad de procesamiento, de almacenamiento y de transferencia de grandes cantidades de datos. A la hora de procesar grandes cantidades de datos, es en este componente donde se ven más claras las soluciones en la nube.

BigQuery

Es un servicio data warehouse gestionado para el análisis sobre datos a gran escala. Permite realizar el análisis de grandes conjuntos de datos a través de consultas SQL super rápidas utilizando el poder del procesamiento de la infraestructura de Google.

Cloud Dataflow

Servicio para el procesamiento de datos por lotes y transmisión en tiempo real.

Cloud Dataproc

Es un servicio gestionado de Spark y Hadoop para procesar grandes conjuntos de datos (procesamiento por lotes, consultas y aprendizaje automático). Permite crear clusteres rápidamente y gestionarlo fácilmente.

Cloud Datalab

Es una herramienta interactiva para la gestión, el análisis y la visualización de datos a gran escala. Asimismo, esta herramienta permite generar modelos de Aprendizaje Automático.

Cloud Dataprep (beta)

Es un servicio para explorar visualmente, limpiar y preparar los datos estructurados (formatos de tablas relacionales) y no estructurados (CSV, JSON) para su posterior análisis. Dataprep detecta automáticamente esquemas, posibles tipos de datos y anomalías de datos.

Pub/Sub de Cloud

Servicio global de mensajería confiable en tiempo real y streaming de datos. Permite enviar mensajes entre aplicaciones. Está diseñado para proporcionar mensajes síncronos, fiables y con varios remitentes y destinatarios entre aplicaciones.

Genomics

Servicio para el procesamiento de datos genéticos de hasta Petabytes. Provee una infraestructura para necesidades bioinformáticas que incluye herramientas para almacenar (Cloud Storage), procesar (Genomics), explorar (BigQuery) y compartir conjunto de datos grandes y complejos de una manera segura. Igualmente implementa Genomics API, un estándar abierto de la Alianza Mundial para la Genómica y Salud.

Google Data Studio(beta)

Permite crear cuadros de mando e informes visualmente atractivos. Se conecta a una variedad de fuentes de datos como BigQuery, Cloud SQL y MySQL. También dispone de funcionalidades para compartir y colaborar con los demás del mismo modo que en Google Drive.

6.3.5 Internet de las Cosas

El componente de Internet de las Cosas ofrece “Cloud IoT Core”, una solución para el Internet de las Cosas que permite conectar millones de dispositivos en la nube.

Cloud IoT Core (Private beta)

Es un conjunto completo de servicios integrados que permite conectar, administrar e ingerir datos a gran escala a partir de dispositivos dispersos por el mundo.

Las principales características del Cloud IoT Core incluyen:

- Uso del protocolo MQTT estándar de la industria.
- Administrador de dispositivos
- Servicio de autenticación de dispositivos
- Control de dispositivos
- Para dispositivos basados en Android Things, las actualizaciones de firmware pueden ser automáticas.
- Balance de carga automática y escala horizontal.
- Seguridad completa gracias a la autenticación basada en certificados y a TLS.
- Integración nativa con los servicios de Google Cloud. Transfiere los datos de IoT entre los servicios de Google Cloud. Ingiere datos con Cloud IoT Core, estos datos se distribuyen a través de Cloud Pub/Sub, luego se aplica transformaciones de datos gracias a Cloud Dataflow y finalmente se pueden almacenar en los servicios de Cloud Storage, Cloud Bigtable o Cloud Spanner.
- Análisis de datos avanzados. Lleva a cabo un análisis *ad hoc* con Google BigQuery, visualiza los datos con Cloud Data Studio y deriva la inteligencia con Cloud Machine Learning.

6.3.6 Aprendizaje Automático

El componente de Aprendizaje Automático ofrece servicios de aprendizaje automático con TensorFlow. Y una colección de API para el análisis de imágenes, conversión de habla en texto, análisis de textos escritos, y traducción de textos.

Cloud Machine Learning Engine (beta)

Es un servicio para desarrollar modelos de aprendizaje automático. Permite hacer principalmente dos cosas:

- Crear modelos de aprendizaje automático utilizando TensorFlow, que es una biblioteca de Software de código abierto creado por Google para la inteligencia

- de máquina.
- Hospedar modelos entrenados para obtener predicciones sobre nuevos datos.

Además, gracias a la integración de este servicio en Cloud Dataflow para llevar a cabo tareas previas al procesado, se puede acceder a los datos ubicados en otros servicios, como Cloud Storage o BigQuery.

API de Cloud Jobs (alfa)

La API de Cloud Jobs provee a las empresas con herramientas para buscar, encontrar y recomendar a candidatos usando por supuesto Machine Learning.

API de Cloud Natural Language

La API de Cloud Natural Language permite analizar la estructura y el significado del texto mediante modelos de aprendizaje automático. Puede utilizarse para extraer información sobre personas, lugares, eventos y muchos elementos que se mencionen en documentos de texto, artículos de noticias, correos electrónicos o redes sociales. De igual forma permite clasificar las palabras en entidades, determinar el nivel de sentimiento y analizar la sintaxis de las oraciones.

API de Cloud Speech

La API de Cloud Speech permite que los desarrolladores conviertan audio en texto, además es capaz de reconocer más de 80 idiomas. Puede transmitir resultados de texto conforme vaya reconociendo el audio. Entre otras características el servicio es capaz de procesar audio en ambientes ruidosos.

API de Cloud Translation

La API de Cloud Translation ofrece una interfaz sencilla para traducir en cualquier idioma admitido. La API es con más de cien idiomas diferentes. La API puede identificar automáticamente un idioma con gran precisión. La integración del API de Cloud Translation es posible a través de una API REST estándar de Google.

API de Cloud Vision

La API Cloud Vision ofrece a los desarrolladores una interfaz sencilla para escanear cualquier imagen y detectar contenidos dentro de estas. La API permite: clasificar imágenes en miles de categorías, detectar objetos y caras, detección de logotipos muy

conocidos, reconocimiento óptico de caracteres y obtención de atributos generales de la imagen. El acceso se lo realiza mediante una API REST.

API de Cloud Video Intelligence (beta)

La API de Cloud Video Intelligence es una API para buscar y descubrir metadatos dentro de un video. Permite:

- La detección de objetos dentro del video tales como perros, plantas, personas, entre otros.
- Detectar cambios de escenas en el video.
- Integración con Google Cloud Platform es a través de una API REST.

6.3.7 Identidad y Seguridad

El componente de Identidad y Seguridad ofrece servicios relacionados con la verificación de la identidad del usuario y la seguridad de cada proyecto de Google Cloud Platform. Permite gestionar el acceso de los usuarios a la nube, gestionar los datos sensibles de una organización, seguridad mediante dispositivos físicos para iniciar sesión, administrador de claves alojados en la nube entre otros. Estos mismos se pueden consultar en el Anexo B.

6.4 Implementación de servicios en Google Cloud Platform

En esta sección se implementarán tres ejemplos para entender el funcionamiento de los servicios Pub/Sub, DataFlow y BigQuery ya que los mismo serán utilizados en el caso de estudio del presente trabajo.

6.4.1 Requisitos

Los siguientes requisitos son necesarios para la ejecución de los ejemplos:

- Tener una cuenta de correo electrónico en Gmail
- Tener un proyecto Google Cloud Platform.
- Tener configurado eclipse para el uso de Google Cloud Platform
- Habilitar los servicios API Pub/Sub, DataFlow y BigQuery.

6.4.2 Publicación y Suscripción de mensajes en Cloud Pub/Sub

En este ejemplo se establecerá la arquitectura de la Figura 18. Se crearán dos clientes: un publicador y un suscriptor. El publicador “publicará” un mensaje en el tema “temaSensor” para luego ser entregado al suscriptor. El envío y la recepción de los mensajes se realizará vía consola de comandos.

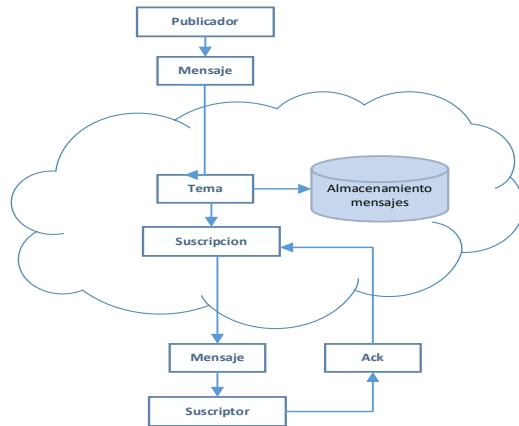


Figura 18: Arquitectura del servicio Pub/Sub.

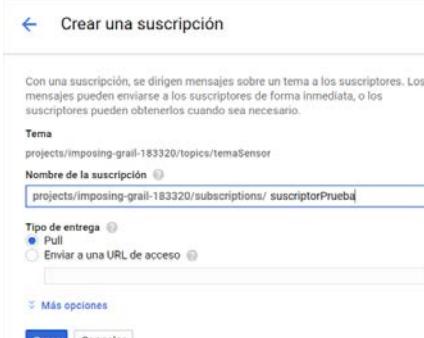
Creación de un tema

La creación de un tema se puede efectuar desde la consola de línea de comando. Pero para el presente ejemplo se realizará desde la plataforma de Google Cloud Platform. Solo la publicación de mensajes y suscripción se realizará desde la consola de comandos. El ingreso a la plataforma se efectuará mediante la url: <https://console.cloud.google.com>.

Una vez dentro, debe dirigirse a la sección Big Data y seleccionar el servicio Pub/Sub.

Dentro del servicio Pub/Sub, para crear el tema y el suscriptor debe seguir los pasos indicados en la Tabla 16.

Tabla 16: Creación de un tema y un suscriptor

Creación del tema	Creación de la suscripción
	

La publicación de mensaje en el tema anteriormente creado, se realizará desde el Shell de Google Cloud Platform, iniciándose el Shell desde la consola del mencionado Google Cloud Platform. Para la publicación de un mensaje en el tema “temaSensor” deben seguirse los pasos expuestos en la Tabla 17.

Tabla 17: Pasos para la publicación de un mensaje desde consola.

Opción para iniciar el Shell	Inicio del servicio desde consola.
	
	
	

6.4.3 Contar las palabras de los libros de Shakespeare en Cloud Dataflow

En el siguiente ejemplo, se contarán las palabras de todos los libros del escritor Inglés William Shakespeare. La fuente de datos (los libros), se encuentran almacenados

en el servicio de ejemplos Google Storage. Estos libros se leerán mediante un proceso DataFlow para realizar la “Extracción, Transformación y Carga” y depositarlo en nuestro servicio Cloud Storage tal como se observa en la Figura 19. El procesamiento de datos se realizará mediante un programa implementado en Java.

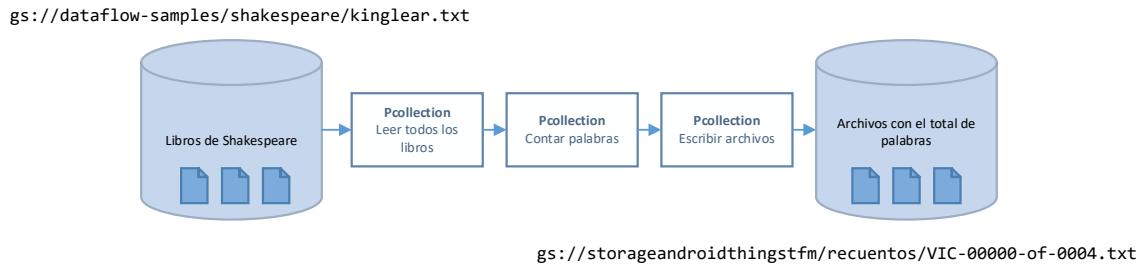


Figura 19: Proceso de extracción, transformación y carga.

Implementación del programa contador de palabras

Para contar las palabras de todos los libros de Shakespeare, se implementó la Clase WordCount. La clase contiene métodos que permiten transformar cada línea de texto de los libros de Shakespeare en una colección de datos. Para el ejemplo, solo se explicará el método principal de la clase implementada. Si alguna vez, llegado el caso, se necesitase descargar el código fuente, se puede realizar desde el vínculo de GitHub [61].

El método principal de la clase WordCount, tiene como misión conectarse a una fuente de datos, y una vez conectado a dicha fuente, recupera la información de los libros. Una vez recuperada la información de los libros, procede a transformarlos en archivos de texto y almacenarlos en un servicio Cloud Storage. La solución se expone en el fragmento de Código 7. En las líneas 4 y 7 se realiza la conexión a la fuente de datos. En la línea 9, se crea un pipeline para almacenar las cadenas de texto de cada libro. En la línea 11, se recuperan las cadenas de palabras de cada libro. En la línea 12, en la variable “p” se obtiene una colección de datos que contiene todas las palabras de cada libro y el total de veces en las que se repite cada palabra. En la línea 13, se transforma la colección de datos obtenida en el paso anterior en archivos de texto. Finalmente, en la línea 14 se inicia la escritura de los archivos de texto en el servicio Cloud Storage.

```

1. public class WordCount {
2.
3.     public static void main(String[] args) {
4.         WordCountOptions options = PipelineOptionsFactory
5.             .fromArgs(args)
6.             .withValidation()
7.             .as(WordCountOptions.class);
8.
9.         Pipeline p = Pipeline.create(options);
10.
11.        p.apply(TextIO.Read.named("ReadLines").from(options.getInputFile()))
12.            .apply(new CountWords())
13.            .apply(MapElements.via(new FormatAsTextFn()))
14.            .apply(TextIO.Write.named("WriteCounts").to(options.getOutput()));
15.
16.        p.run();
17.    }
18. }

```

Código 7: Programa contador de palabras.

Ejecución del programa contador de palabras

Para ejecutar el programa desde Eclipse, en el menú “Run” debe elegirse la opción “Run Configuración” e ingresar los parámetros de conexión a la nube de Google tal como se observa en la Figura 20.

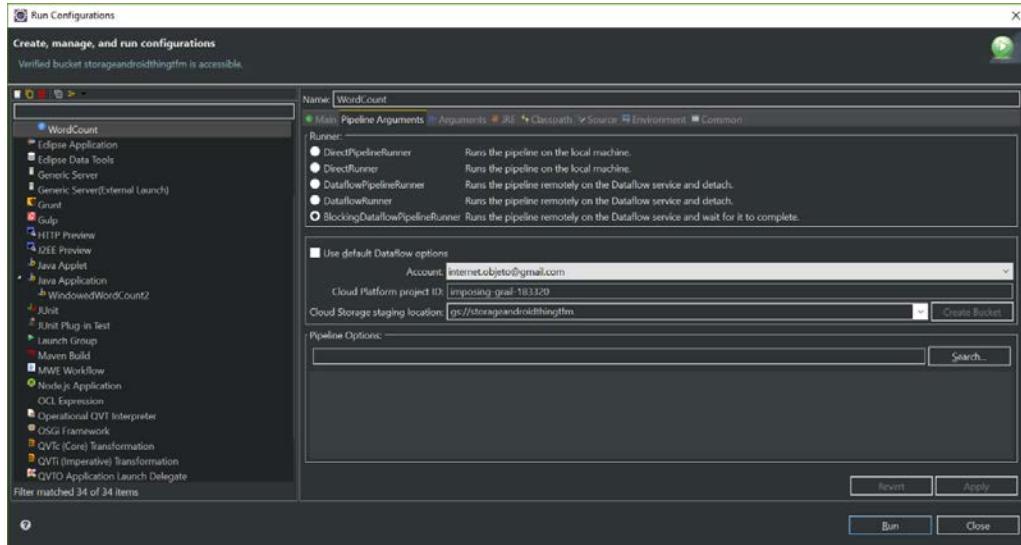


Figura 20: Argumentos de conexión: Cuenta, ID Proyecto y ubicación del Cloud Storage.

Finalmente, el resultado de la ejecución se puede observar desde la consola de Google Cloud Platform, tal como se observa en la Figura 21.

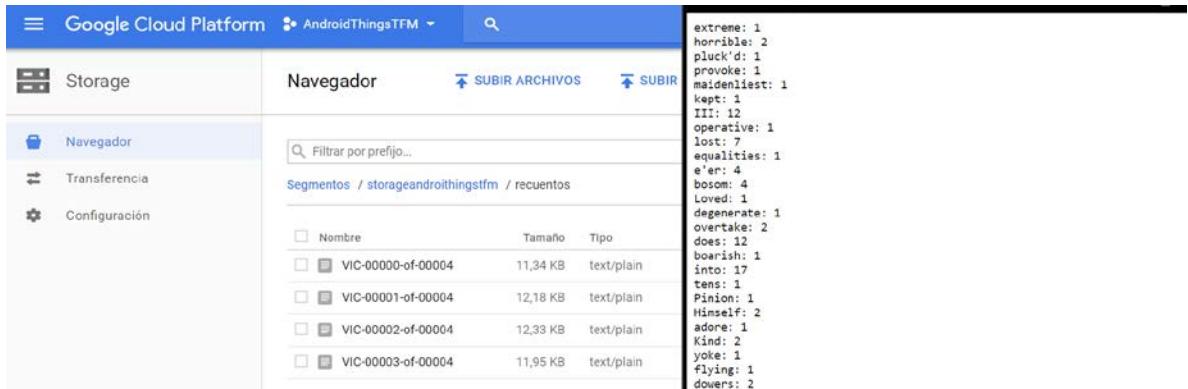


Figura 21: Vista de la ejecución de los procesos de transformación de datos [52].

6.4.4 Consulta del historial revisiones de Wikipedia en BigQuery

En este ejemplo, se probará la potencia en procesamiento del servicio BigQuery, para ello se realizará una consulta del historial completo de revisiones de todos los artículos de Wikipedia hasta abril del 2010, con el propósito de conocer el total de registros generados en la revisión de artículos desde su creación.

Para la ejecución de la consulta, se debe seleccionar del menú de la derecha, la opción BigQuery tal como se aprecia en la Figura 22.

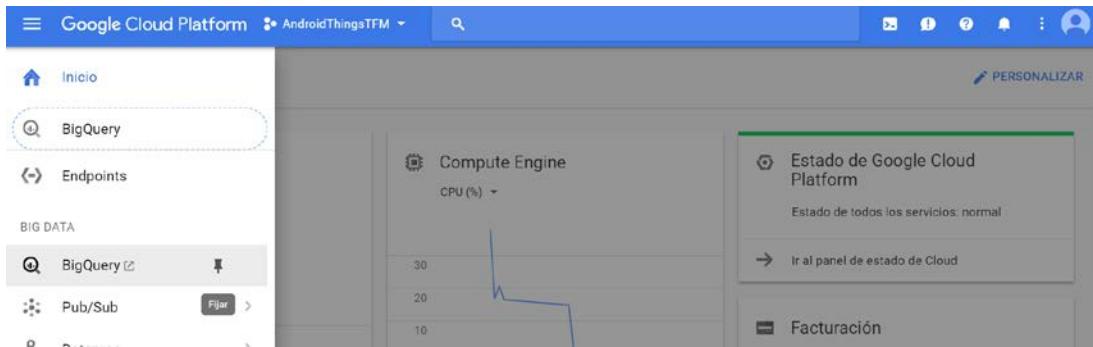
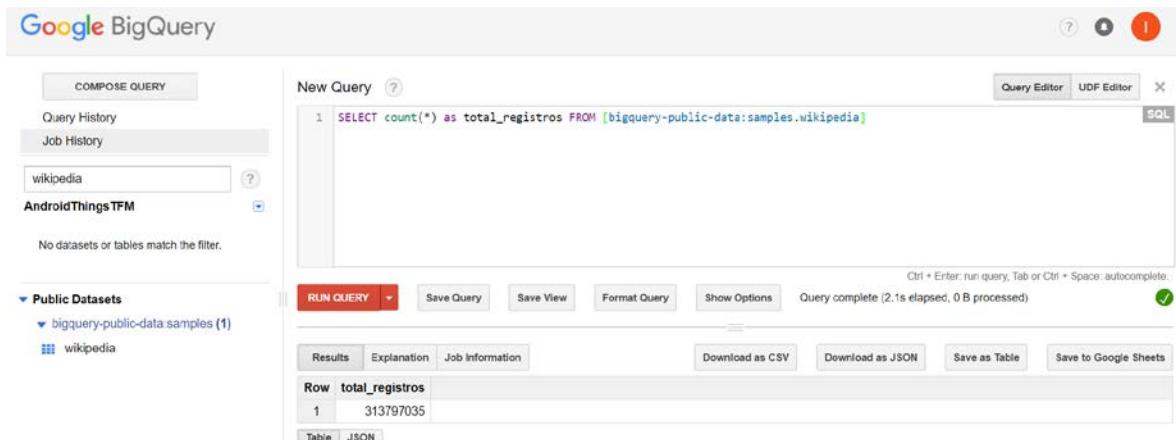


Figura 22: Menú de acceso al servicio de BigQuery [52].

En la Figura 23, se observa el resultado de una consulta en una tabla de aproximadamente 313 millones de registros en 2.1 segundos.



The screenshot shows the Google BigQuery web interface. On the left, there's a sidebar with 'COMPOSE QUERY' and sections for 'Query History', 'Job History', and datasets like 'wikipedia' and 'AndroidThingsTFM'. Below these, under 'Public Datasets', are 'bigquery-public-data.samples (1)' and 'wikipedia'. A message says 'No datasets or tables match the filter.' In the main area, a 'New Query' window is open with the SQL query:

```
1 SELECT count(*) as total_registros FROM [bigquery-public-data:samples.wikipedia]
```

The status bar at the bottom right indicates 'Query complete (2.1s elapsed, 0 B processed)' with a green checkmark. Below the query window, there are tabs for 'Results', 'Explanation', and 'Job Information'. Under 'Results', a single row is shown:

Row	total_registros
1	313797035

There are also buttons for 'Download as CSV', 'Download as JSON', 'Save as Table', and 'Save to Google Sheets'.

Figura 23: Resultado de la ejecución de una consulta BigQuery [52].

Capítulo 7

Desarrollo de un sistema medioambiental basado en IoT

7.1 Descripción del problema

Informes de la Organización Mundial de Salud mencionan que, en los últimos 50 años, la actividad humana y, en particular, el consumo de combustibles fósiles, han hecho que se liberen mayores cantidades de CO₂ y otros gases a la atmósfera alterando el clima mundial. La alteración del clima, ha permitido que se desate una amplia variedad de consecuencias, por ejemplo: aumento del nivel del mar, tormentas, sequías, incendios entre ellas la salud humana.

En relación a la salud humana, en la literatura existen varios estudios relacionados con los efectos medioambientales, y más específicamente con los efectos de la temperatura y presión atmosférica sobre la salud de las personas.

Por ejemplo, Castro [11] estudió las incidencias de las hemorragias subaracnoidea aneurismática y la influencia de los cambios de la presión atmosférica en los pacientes ingresados en el servicio de neurocirugía del Hospital Antonio Lenin Fonseca de Nicaragua. Los resultados que obtuvo en relación a la presión atmosférica y la temperatura a la cual los pacientes se expusieron, destacaron una mayor incidencia de hemorragia subaracnoideas en aquellos que provenían de regiones donde se presentaban mayores cambios de la presión atmosférica y la temperatura y observó una menor incidencia en aquellos otros pacientes procedentes de regiones donde hubo una menor variación climatológica de la temperatura ambiental y presión atmosférica.

Del mismo modo, Gaona [12] evaluó la influencia de las variables meteorológicas con respecto a la movilidad respiratoria en niños de diferentes hogares distribuidos en la ciudad de Sao Paulo (Brasil) donde estudió los eventos de sibilancia (enfermedad respiratoria) relacionados con las condiciones de la calidad del aire y las condiciones meteorológicas externas. Los resultados que obtuvo señalaron que las variaciones de presión que generan frentes fríos y olas de calor son los posibles factores de riesgo para el desarrollado de la sibilancia.

Otro estudio dentro de la línea que aquí analizamos, lo realizó Herrera [16], quien efectúo una encuesta a campesinos del centro de Santander(Colombia) con el fin de conocer sus percepciones sobre los fenómenos de variabilidad climática y cambios climáticos. Sus hallazgos demostraron que la variabilidad climática es un tema muy reconocido y que la gente explica bien los cambios en el clima regional, sus causas e impactos.

Asimismo, Olcina [13] analizó el efecto de las variaciones en la densidad del oxígeno, originadas por los cambios en las masas de aire presentes en la atmósfera, sobre los ingresos hospitalarios por enfermedad cardiaca y pulmonares, registrados en la ciudad de Alicante (España). Los resultados obtenidos destacaron que una situación atmosférica que favorezca a una mayor o menor densidad de oxígeno, no es causa directa del desarrollo de dolencias, pero que sí puede actuar como elemento favorecedor de su desarrollo en grupos de riesgo (personas mayores de 65 años) y personas con afección previa de dichas enfermedades. Destaca también el hecho de que las enfermedades cardíacas muestran una estacionalidad marcada, siendo los tres primeros y los tres últimos meses del año los que recogen mayor número de ingresos, coincidiendo con los meses más fríos.

Igualmente, Rebolledo [17] estudió las respuestas fisiológicas (mareos, somnolencia, náuseas, dolor de cabeza, malestar estomacal conducentes a vómitos y diarrea, etc.) a los cambios atmosféricos en turistas que visitan zonas de altura en Chile. Encontró que las respuestas fisiológicas están relacionadas con la enfermedad denominada “Mal Agudo de Montaña” (MAM). En su artículo, recomienda ciertos cuidados para mejorar el proceso de aclimatación y disminuir el MAM. Entre los cuidados menciona algunos como: realizar ascensos progresivos, evitar las comidas pesadas, mantenerse hidratado, detener el ascenso en cuanto aparezcan síntomas entre otros.

De la misma forma, Paolaso [14] en su estudio de la Biometeorología, encontró factores relacionados con la composición de las masas de aire, la interacción entre la temperatura, presión atmosférica y humedad. En sus resultados menciona que la baja presión atmosférica pudiera tener vinculación con la mala conducta de los escolares, y con algunos trastornos patológicos como el incremento de infartos cardíacos y úlceras sangrantes. También, identificó posibles síntomas psíquicos tales como el malestar general, malhumor, tristeza, decaimiento o falta de fuerzas, irritabilidad y mutismo que se producen cuando baja la presión atmosférica.

Como acabamos de analizar, existen varias investigaciones relacionadas con los efectos medioambientales, específicamente los relacionados con la temperatura y la

presión atmosférica sobre la salud de las personas. Entre los diferentes efectos mencionados, podemos destacar un problema bastante cotidiano: la percepción de dolores por algunas personas en su cuerpo y o los cambios de conducta justo cuando hay un evento climático. No se sabe a ciencia exacta a qué se debe, sin embargo, estudios lo relacionan a los cambios climáticos tal como se comentó en párrafos anteriores.

Partiendo del anterior problema, entre las múltiples causas [11], [12], [16], [13], [17], [14], [15] que puede originar este inconveniente hallamos las siguientes:

- Fenómenos climatológicos inexplicables.
- Contaminación del medioambiente.
- Masas de aire caliente o frío.
- Épocas del año.
- Presencia de oxígeno en el aire
- Cambios constantes de temperie.
- Habitad humano.
- Factores genéticos
- Edad
- Alimentación
- Etc.

Las consecuencias [11], [12], [16], [13], [17], [14], [15] que puede traer el problema de los dolores en el cuerpo y los cambios de conducta son muchas, entre las que destacamos a continuación:

- Disminución de la productividad.
- Salud mental.
- Cefaleas.
- Equilibrio humoral.
- Actividad nerviosa.
- Aumento de enfermedades.
- Enfermedades cardíacas.
- Enfermedades pulmonares.
- Enfermedades de presión arterial (alta y baja)
- Muerte.
- Etc.

7.2 Caso de estudio TEMPRE: Sistema de apoyo para el estudio del efecto medioambiental sobre salud de las personas.

Una posible solución al problema desde el punto de vista tecnológico, pasa por implementar un sistema medioambiental que permita recolectar datos de la temperatura y presión atmosférica para que, a partir de estos, el sistema pueda analizar e identificar patrones de datos que afectan a la salud de las personas. La solución a nivel global se plantea en la Figura 24, en ella se observan Sistemas Empotrados encargados de la recolección de datos medioambientales, una plataforma de computación en la nube encargada del procesamiento, análisis y gestión de los datos generados por los Sistemas Empotrados y un dispositivo móvil con una aplicación desarrollada en Android para retroalimentar el sistema con la información generada por el usuario y así en un futuro, el sistema medioambiental identifique patrones de datos que afecten al estado de salud de las personas.

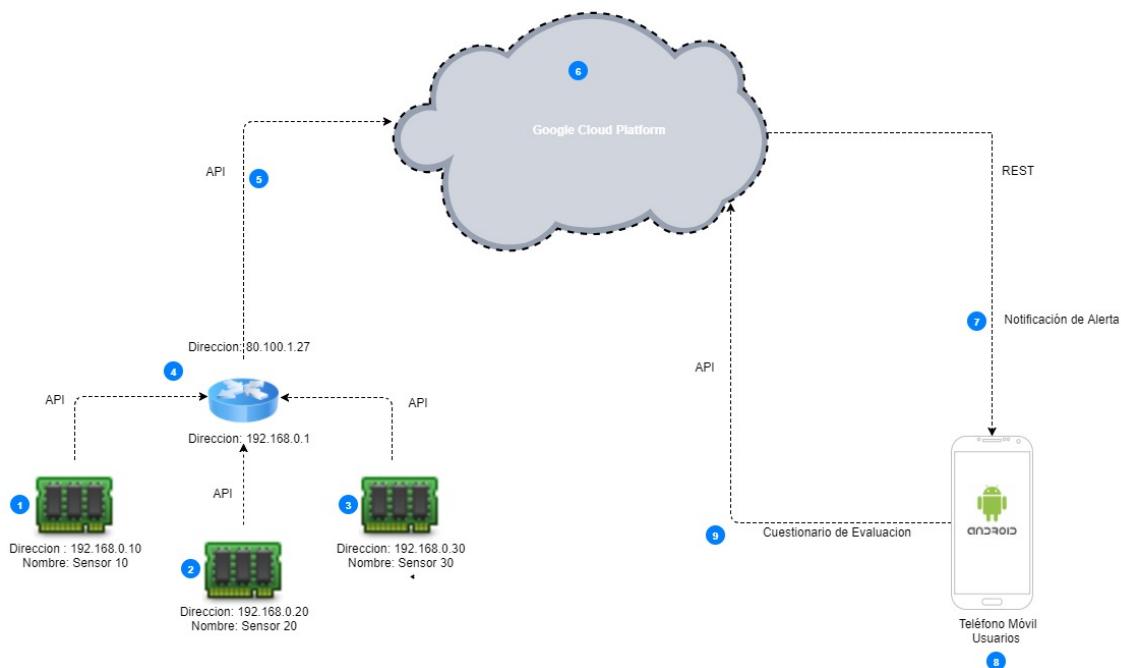


Figura 24: Arquitectura global del Sistema Medioambiental.

7.3 Arquitectura

La arquitectura de la Figura 25 propuesta para la implementación del sistema medioambiental, se encuentra dividida en tres capas: la Capa de Detección, la Capa de Computación en la Nube y Capa de Aplicación.

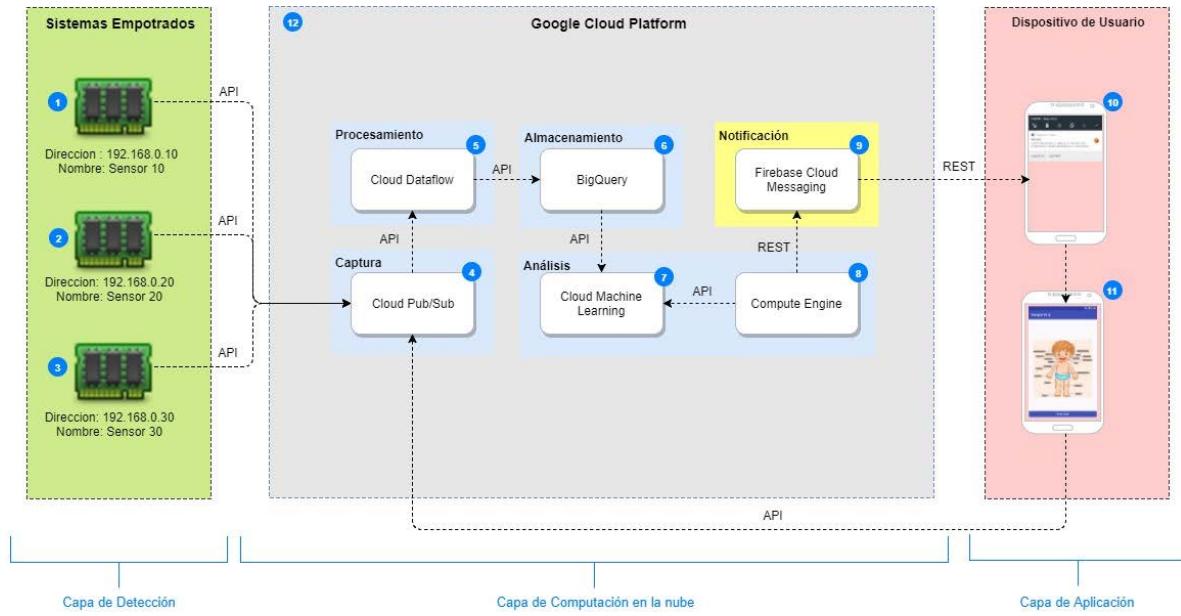


Figura 25: Arquitectura en capas del Sistema Medioambiental.

7.3.1 Capa de Detección

La Capa de Detección es la que permitirá recolectar la información medioambiental a través de Sistemas Empotrados. Los Sistemas Empotrados serán los encargados de la recolección de la información del entorno. Estos estarán compuestos de plataformas de Hardware Raspberry PI y placas de circuitos modulares Rainbow HAT. Las placas de circuitos modulares Rainbow HAT (son “shields” que incluyen sensores de temperatura y presión) serán montados sobre la Raspberry PI. El manejo de los recursos del Hardware de los Sistemas Empotrados será gestionado por el sistema operativo Android Things y por su parte, la programación de la aplicación que funcionará en Android Things se realizará con el lenguaje de alto nivel Java. El envío de los datos a la Capa de Computación en la nube se ejecutará mediante la API de Google Cloud Platform.

7.3.2 Capa de Computación en la nube

La Capa de Computación en la nube es la que permitirá capturar, procesar, almacenar y analizar la información recolectada en la Capa de Detección, para ello se utilizará: el servicio Cloud Pub/Sub, Cloud DataFlow, BigQuery, Cloud Machine Learning, Compute Engine y Firebase Cloud Messaging.

El servicio Cloud Pub/Sub se utilizará para capturar los datos enviados por los Sistemas Empotrados.

El servicio Cloud DataFlow permitirá mover en tiempo real los datos recolectados en el Cloud Pub/Sub hacia el servicio de almacenamiento de datos BigQuery.

Como bien se mencionó anteriormente, BigQuery es un servicio de almacenamiento de datos a gran escala, no volátil y se utilizará para la acumulación de los datos.

Para que el sistema encuentre patrones en los datos almacenados se utilizará el servicio de Cloud Machine Learning, principalmente para dos objetivos: crear modelos de aprendizaje automático utilizando TensorFlow y hospedar modelos entrenados para obtener predicciones sobre nuevos datos.

El servicio Compute Engine contendrá un servidor Linux que a través de una API se conectará al servicio Cloud Machine Learning para obtener las predicciones y enviar estas a los teléfonos móviles de cada usuario, por medio de notificaciones push utilizando el servicio Firebase Cloud Messaging.

El servicio de mensajería Firebase Cloud Messaging será el encargado de comunicar a los clientes de la existencia de un nuevo mensaje. Los mensajes generados en el servicio Compute Engine serán entregados como notificaciones a los teléfonos móviles de cada usuario por medio del servicio Firebase Cloud Messaging.

7.3.3 Capa de Aplicación

La Capa de Aplicación es la que se encarga de interactuar con los usuarios a través de notificaciones y cuestionarios. Para realizar esta tarea se utilizarán aplicaciones desarrolladas en Android. Las notificaciones se entregarán por medio de la aplicación Android instalados en los teléfonos móviles de cada usuario, estas notificaciones serán entregadas por el servicio Firebase Cloud Messaging. Los cuestionarios se llenarán con

la misma aplicación del usuario, para posteriormente enviar los datos al servicio Cloud Pub/Sub con el objetivo de que estos cuestionarios se almacenen en una tabla de BigQuery para su posterior análisis.

7.4 Requisitos de Hardware y Software

Los requisitos de Hardware y Software que se utilizarán en la implementación del sistema medioambiental propuesto serán los siguientes:

7.4.1.1 Hardware

- Raspberry PI 3
- Rainbow HAT

7.4.1.2 Software

- Android Studio 2.3.3
- Imagen Android Things preview 0.4.1
- Eclipse Neon 2 Release 4.6.2
- Google Cloud SDK Shell
- SDFormatter V4.0
- Win32 Disk Imagen Version 0.9.5
- Bibliotecas en Android Studio 2.3.3 (Gradle)
 - androidthings:0.4-devpreview
 - driver-button:0.3
 - driver-bmx280:0.2
 - driver-ht16k33:0.3
 - driver-apa102:0.3
 - driver-pwmspeaker:0.2
 - google-api-client-android:1.22.0
 - google-api-services-pubsub:v1-rev12-1.22.0
- Bibliotecas en Eclipse Neon 2 Release 4.6.2 (Maven)
 - jackson-core 2.9.1
 - google-http-client 1.22.0
 - google-api-client 1.22.0
 - google-cloud-dataflow-java-sdk-parent 1.9.1
 - google-api-services-dataflow v1b3-rev43-1.22.0

- o google-api-services-bigquery v2-rev295-1.22.0

7.5 Desarrollo de la Capa de Detección

La implementación del sistema medioambiental se efectuó en capas, tal como se propuso en la sección 7.3. En esta sección se explicará el desarrollo de los elementos que integran la Capa de Detección que es la que permite recolectar la información medioambiental por medio de Sistemas Empotrados, de manera similar a la mostrada en la Figura 26.



Figura 26: Capa de Detección.

Para la construcción de esta capa fue necesario llevar a cabo los siguientes pasos:

- Ensamblado del Hardware.
- Instalación de Android Things en Raspberry Pi.
- Conexión entre la computadora y el Sistema Empotrado.
- Creación de la aplicación Android Things.
- Despliegue de la aplicación Android Things.

7.5.1 Ensamblado del Hardware

El ensamblado del Hardware se realizó como se observa en la Figura 27 y se utilizaron los siguientes elementos:

- Raspberry Pi
- Rainbow HAT
- Cable ethernet
- Cable Usb macho-macho
- Cable HDMI
- Teclado Usb
- Mouse Usb
- Monitor LCD

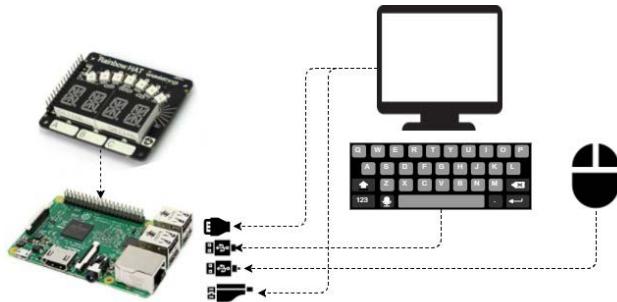


Figura 27: Ensamblado del Hardware.

El ensamblado del Sistema Empotrado consistió en montar el módulo Rainbow HAT sobre Raspberry PI y descargar los controladores para la comunicación con los sensores.

Los cables permitieron la conexión entre la computadora y Sistema Empotrado y esta conexión permitió efectuar el enlace con el IDE Android Studio y desplegar la aplicación Android Things a través de cable ethernet hacia el Sistema Empotrado.

Los periféricos de entrada y salida (teclado, mouse y monitor lcd) se utilizaron para navegar en el Sistema Operativo Android Things y ver la información de la conexión a la red, como número de IP, que son necesarios para realizar la conexión entre el Sistema Empotrado y el IDE Android Studio.

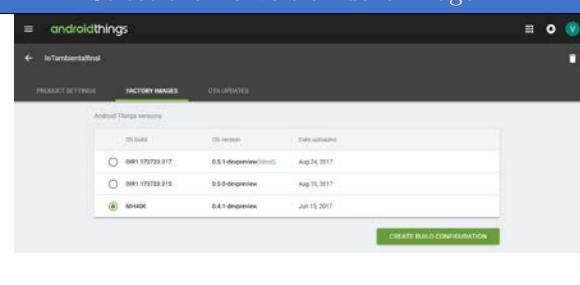
7.5.2 Instalación de Android Things en Raspberry Pi

La instalación de Android Things en Raspberry Pi conllevó los siguientes pasos:

- Descarga la imagen de Android Things.
- Insertar la tarjeta SD en la computadora.
- Formatear la tarjeta SD.
- Descomprimir la imagen de Android Things en la tarjeta SD.
- Sacar la tarjeta SD e insertar en la Raspberry PI

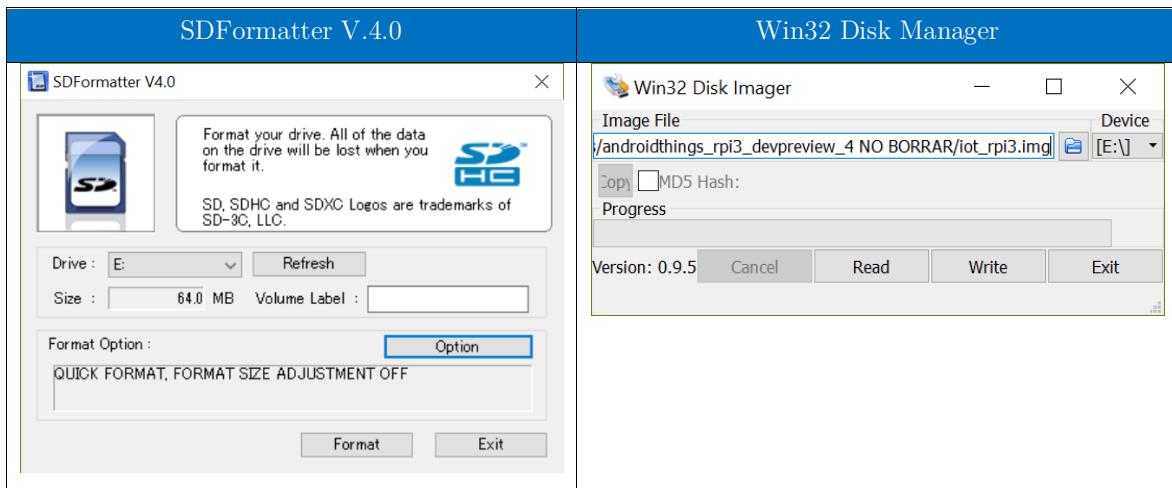
La descarga de la imagen de Android Things se realizó desde el vínculo [39], los pasos que se siguieron para la descarga de la imagen desde la consola de desarrolladores Android Things, se indican en la Tabla 18.

Tabla 18: Descarga de la imagen de Android Things.

Crear un proyecto de referencia Android Things	Seleccionar la versión de la imagen
	

Terminada la descarga de la imagen de Android Things, el siguiente paso fue la descompresión de la misma, aunque antes fue necesario formatear la tarjeta SD, para lo que se utilizó la aplicación SDFormater V4.0. Después de formatear la tarjeta SD se procedió a descomprimir la imagen del Sistema Operativo Android Things, utilizando la aplicación Wind32 Disk Manager, así como se observa en la Tabla 19.

Tabla 19: Formateado del SD y descompresión de la imagen de Android Things.



7.5.3 Conexión entre la computadora y el Sistema Empotrado.

Una vez terminada la descompresión de la imagen de Android Things en la tarjeta SD, la siguiente tarea fue vincular la computadora con el Sistema Empotrado. Para realizar esta tarea de vinculación fue necesario:

- Obtener la dirección IP del Sistema Empotrado.
- Ejecutar el Shell de Android.

Para obtener la dirección IP del Sistema Empotrado, se necesitó encender la Raspberry PI -conectado al monitor- y esperar a que el router asignara, por medio del servidor DHCP, una dirección IP en sus dos versiones IPv4/IPv6, así como se observa en la Figura 28.



Figura 28: Visualización de la dirección IP del Raspberry PI.

La IP obtenida permitió establecer conexión entre la computadora y el Sistema Empotrado. La conexión se efectuó por medio del Shell de Android para lo que se ejecutó el comando de la Figura 29, donde fue necesario ingresar el comando “adb connect”, seguido de la dirección IP del Sistema Empotrado y el puerto de comunicación que por defecto es 5555.

```
C:\Users\Vic>adb connect 192.168.137.148
connected to 192.168.137.148:5555
C:\Users\Vic>
```

Figura 29: Comando de conexión adb.

7.5.4 Creación de la aplicación Android Things

La aplicación que recolecta la información medioambiental fue programada en Java utilizando el IDE Android Studio y el SDK 24. Con la aplicación implementada, se tuvo acceso a los periféricos de entrada y salida del Raspberry PI, el cual permitió leer las medidas de temperatura y presión atmosférica obtenidas por el módulo Rainbow HAT. Según la documentación oficial para desarrolladores, Android Things es compatible con aplicaciones desarrolladas con el SDK nivel 24 o superior.

En la Figura 30, se observan los directorios y archivos que conforman el proyecto de la aplicación Android Things.

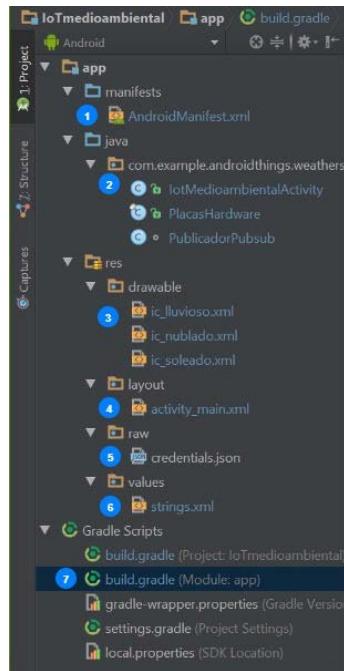


Figura 30: Estructura de la aplicación Android Things.

De toda la lista de archivos en las siguientes secciones se explicarán fragmentos de códigos de los archivos más relevantes y estos son:

- AndroidManiest.xml
- iotMediombientalActivity.java
- PlacasHardaware.java
- PublicadorPubsub.java
- activity_main.xml
- credentials.json
- build.gradle (Modulo app)

AndroidManifist.xml

Situado en la raíz del proyecto como AndroidManifiest.xml, es un archivo que permite aplicar configuraciones a una aplicación Android Things. Dentro del archivo Android Manifiest.xml se declara todo lo que pueda suceder en una aplicación, tales como: permisos para acceder a la red, a internet, a los recursos de Hardware, a servicios, etc.

En el caso de la aplicación medioambiental, en el fragmento de Código 8, en las líneas 5 y 6 se otorgaron permisos para acceder a Internet y a la red. En las líneas 7 y 8 se concedieron permisos a los controladores de Android Things.

```
1. <?xml version="1.0" encoding="utf-8"?>
2. <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3.   package="com.example.androidthings.weatherstation">
4.
5.   <uses-permission android:name="android.permission.INTERNET" />
6.   <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
7.   <uses-
8.     permission android:name="com.google.android.things.permission.MANAGE_INPUT_DRIVER
9.     S" />
10.    <uses-
11.      permission android:name="com.google.android.things.permission.MANAGE_SENSOR_DRIVE
12.      RS" />
13.    <application>
14.    ...
15.    </application>
16.  </manifest>
```

Código 8: AndroidManifest.xml.

IoTMedioambientActivity.java

El archivo IoTMedioambientActivity.java es la clase principal de la aplicación, en esta clase se definen los controladores, métodos, así como el funcionamiento del layout activity_main.xml.

En el fragmento de Código 9, correspondiente a la clase IoTMedioambientActivity.java, los datos de los sensores fueron obtenidos por medio del objeto mSensorManager declarado en la línea 10. Para establecer la comunicación con el módulo Rainbow HAT se utilizaron los controladores declarados en las líneas 12 al 16.

```
1. public class IotMedioambientActivity extends Activity {
2.
3.     private static final String TAG = IotMedioambientActivity.class.getSimpleName();
4.
5.     private enum DisplayMode {
6.         TEMPERATURA,
7.         PRESION
8.     }
9.
10.    private SensorManager mSensorManager;
11.}
```

```

12.    private ButtonInputDriver mButtonInputDriver;//boton touch
13.    private Bmx280SensorDriver mEnvironmentalSensorDriver;//Sensor de Temperatura
y presión
14.    private AlphanumericDisplay mDisplay; //Display de 14 segmentos
15.    private Apa102 mLedstrip;//Led de colores
16.    private int[] mRainbow = new int[7];
17.    private DisplayMode mDisplayStyle = DisplayMode.TEMPERATURA;
Código 9: IoTMedioambientalActivity.java - objeto SensorManager y Controladores

```

El fragmento de Código 10, es el encargado de obtener la medida de la temperatura. Para obtener la medida de la temperatura, se implementó el método onSensorChanged quien recupera constantemente el valor obtenido por el sensor de temperatura y actualiza la pantalla de 14 segmentos del Sistema Empotrado.

```

1. //Devolución de llamada cuando SensorManager entrega datos de temperatura.
2. private SensorEventListener mTemperaturaListener = new SensorEventListener() {
3.     @Override
4.     public void onSensorChanged(SensorEvent event) {
5.         mUltimaTemperatura = event.values[0];
6.         Log.d(TAG, "sensor cambiado: " + mUltimaTemperatura);
7.         if (mDisplayStyle == DisplayMode.TEMPERATURA) {
8.             updateDisplay(mUltimaTemperatura);
9.         }
10.    }
11. };

```

Código 10: IoTMedioambientalActivity.java - Obtención de la temperatura.

De forma similar al anterior fragmento de Código 11, permite obtener la medida de la presión atmosférica. El método onSensorChanged es el responsable de leer constantemente el valor obtenido por el sensor de presión atmosférica y actualizar el display de 14 segmentos del Sistema Empotrado.

```

1. // Devolución de llamada cuando SensorManager entrega datos de presión.
2. private SensorEventListener mPresionListener = new SensorEventListener() {
3.     @Override
4.     public void onSensorChanged(SensorEvent event) {
5.         mUltimaPresion = event.values[0];
6.         Log.d(TAG, "sensor cambiado: " + mUltimaPresion);
7.         if (mDisplayStyle == DisplayMode.PRESION) {
8.             updateDisplay(mUltimaPresion);
9.         }
10.    }
11. }
12. };

```

Código 11: IoTMedioambientalActivity.java - Obtención de la presión.

En el fragmento de Código 12, el método onCreate representa el momento en el que la aplicación se crea. Este método normalmente lo genera el asistente al crear una nueva actividad en Android Things, y es donde se creará todo lo que vaya a necesitar la aplicación. En el método onCreate, en la línea 5, se instanció un objeto de la clase SensorManager para permitir acceder a los sensores. En las líneas restantes se iniciaron la conexión a los sensores de temperatura y presión, al display de 14 segmentos, al led, altavoz. Asimismo, en la línea 16 se instanció el objeto handler para iniciar la comunicación con el hilo principal de la aplicación. También dentro de este método se obtuvieron las credenciales para la publicación de la temperatura y presión en la nube de Google Cloud Platform.

```

1.  @Override
2.  protected void onCreate(Bundle savedInstanceState) {
3.      super.onCreate(savedInstanceState);
4.      Log.d(TAG, "Inicio IoT Medioambiental");
5.      mSensorManager = ((SensorManager) getSystemService(SENSOR_SERVICE));
6.      try {
7.          mEnvironmentalSensorDriver = new Bmx280SensorDriver(PlacasHardware.get
I2cBus());
8.          mDisplay = new AlphanumericDisplay(PlacasHardware.getI2cBus());
9.          // Inicializando la tira Led de colores
10.         mLedstrip = new Apa102(PlacasHardware.getSpiBus(), Apa102.Mode.BGR);
11.         // Led
12.         PeripheralManagerService pioService = new PeripheralManagerService();

13.        // Altavoz
14.         mSpeaker = new Speaker(PlacasHardware.getSpeakerPwmPin());
15.        // El objeto handler permite comunicar con el hilo principal de la aplicación
16.         Handler handler = new Handler(getMainLooper());
17.         handler.postDelayed(new Runnable() {
18.             @Override
19.             public void run() {
20.                 slide.start();
21.             }
22.             }, ALTAVOZ_LISTO_RETRASO_MS);
23.         } catch (IOException e) {
24.             throw new RuntimeException("Error al inicializar speaker", e);
25.         }
26.
27.        //Iniciar el Cloud PubSub Publicador si hay credenciales en la nube.
28.         int credentialId = getResources().getIdentifier("credentials", "raw",
getP
ackageName());
29.     }

```

Código 12: IoTMedioambientalActivity.java - Método onCreate.

PlacasHardware.java

El archivo PlacasHardware.java descrito en el fragmento de Código 13, es una clase que permite a la aplicación soportar varios perfiles de Hardware.

```
1. public final class PlacasHardware {
2.     private static final String DISPOSITIVO_EDISON_ARDUINO = "edison_arduino";
3.     private static final String DISPOSITIVO_EDISON = "edison";
4.     private static final String DISPOSITIVO_JOULE = "joule";
5.     private static final String DISPOSITIVO_RPI3 = "rpi3";
6.     private static final String DISPOSITIVO_IMX6UL_PICO = "imx6ul_pico";
7.     private static final String DISPOSITIVO_IMX6UL_VVDN = "imx6ul_iopb";
8.     private static final String DISPOSITIVO_IMX7D_PICO = "imx7d_pico";
9.     private static String sBoardVariant = "";
10.
11.    public static String getButtonGpioPin() {
12.        switch (getBoardVariant()) {
13.            case DISPOSITIVO_EDISON_ARDUINO:
14.                return "IO12";
15.            case DISPOSITIVO_EDISON:
16.                return "GP44";
17.            case DISPOSITIVO_JOULE:
18.                return "J7_71";
19.            case DISPOSITIVO_RPI3:
20.                return "BCM21";
21.            case DISPOSITIVO_IMX6UL_PICO:
22.                return "GPIO2_IO03";
23.            case DISPOSITIVO_IMX6UL_VVDN:
24.                return "GPIO3_IO01";
25.            case DISPOSITIVO_IMX7D_PICO:
26.                return "GPIO_174";
27.            default:
28.                throw new IllegalArgumentException("Dispositivo desconocido: " +
Build.DEVICE);
29.        }
30.    }
}
```

Código 13: PlacasHardware.java.

PublicadorPubSub.java

El archivo PublicadorPubSub.java es una clase que envía los datos de la temperatura y presión atmosférica al servicio de computación en la nube Google Cloud Platform. En esta clase, en la línea 7 del fragmento de Código 14, se definió el intervalo de tiempo para publicar los datos de los sensores en la nube, en la línea 15 se especificó la referencia al “Tema” del servicio Cloud Pub/Sub, en la línea 17 se determinaron las credenciales de conexión con la nube y el formato de los datos enviados; los datos enviados a la nube tienen el formato json.

```

1. class PublicadorPubsub {
2.     private static final String TAG = PublicadorPubsub.class.getSimpleName();
3.
4.     private float mUltimaTemperatura = Float.NaN;
5.     private float mUltimaPresion = Float.NaN;
6.
7.     private static final long PUBLISH_INTERVAL_MS = TimeUnit.MINUTES.toMillis(1);
8.
9.     PublicadorPubsub(Context contexto,
10.                      String nombreAplicacion,
11.                      String proyecto,
12.                      String tema,
13.                      int credentialResourceId) throws IOException {
14.     mContexto = contexto;
15.     mNombreAplicacion = nombreAplicacion;
16.     mTema = "projects/" + proyecto + "/topics/" + tema;
17.     InputStream jsonCredentials = mContexto.getResources().openRawResource(cr
eidentialResourceId);
18.
19.     mHttpTransport = AndroidHttp.newCompatibleTransport();
20.     JsonFactory jsonFactory = JacksonFactory.getDefaultInstance();
21.     mPubsub = new Pubsub.Builder(mHttpTransport, jsonFactory, credentials).
setApplicationName(mNombreAplicacion).build();
22. }

```

Código 14: *PublicadorPubsub.java*.

Activity_main.xml

En el archivo xml Activity_main.xml se definió la interfaz del Sistema Empotrado. El fragmento de Código 15, permitió crear la interfaz visual de la Activity; que prácticamente consistió en una Activity con una imagen. La imagen de la activity cambia según el nivel de la temperatura y presión atmosférica.

```

1. <?xml version="1.0" encoding="utf-8"?>
2. <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3.     android:layout_width="match_parent"
4.     android:layout_height="match_parent"
5.     android:orientation="vertical"
6.     android:padding="16dp">
7.
8.     <ImageView
9.         android:id="@+id/imageView"
10.        android:layout_width="match_parent"
11.        android:layout_height="match_parent"
12.        android:src="@drawable/ic_nublado" />
13. </LinearLayout>

```

Código 15: *main_activity.xml*.

Al final la interfaz de la aplicación quedo tal como se observa en la Figura 31.

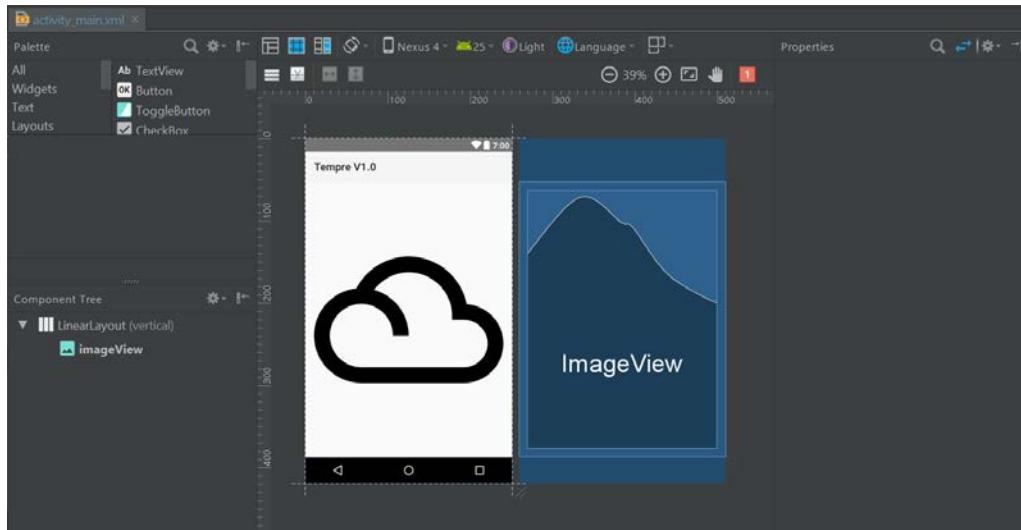


Figura 31: Interfaz de la aplicación.

Credentials.json

El archivo Credentials.json, es un archivo que permite conectar a la aplicación con la nube de Google. El archivo Credenciales.json prácticamente contiene parámetros de conexión hacia la nube de Google tal como se observa en el fragmento de Código 16. La generación e inclusión del archivo Credentials.json se efectuó siguiendo los pasos de la documentación de Google Cloud Platform [62].

```
1. {
2.   "type": "service_account",
3.   "project_id": "androidthingsvic-170713",
4.   "private_key_id": "3c52bdc83ae0000fef96c2434a0dfd77583b21c2",
5.   "private_key": "-----BEGIN PRIVATE KEY-----
6.   \nMIIEvQIBADANBgkqhkiG9w0BAQEFAASCBKcwggSjAgEAAoIBAQCgGZqF+N1S194h\nn2gmj7ybXWqJJA
7.   I
8.   "client_email": "publicadorsensor@androidthingsvic-
9.   170713.iam.gserviceaccount.com",
10.  "client_id": "100855013727554701179",
11.  "auth_uri": "https://accounts.google.com/o/oauth2/auth",
12.  "token_uri": "https://accounts.google.com/o/oauth2/token",
13.  "auth_provider_x509_cert_url": "https://www.googleapis.com/oauth2/v1/certs",
14.  "client_x509_cert_url": "https://www.googleapis.com/robot/v1/metadata/x509/publ
15.  icadorsensor%40androidthingsvic-170713.iam.gserviceaccount.com"
16. }
```

Código 16: Credenciales.json.

Builde.gradle

Build.gradle es un archivo de construcción y gestión de la aplicación, dentro del cual, entre las líneas 4 y 12 del fragmento de Código 17, se definieron las versiones del sdk compatibles con Android Things. En las líneas 16 y 17 se establecieron los parámetros de conexión a la nube de Google. Por su parte, entre las líneas 22 a 28 se incluyeron las bibliotecas del frameworks y las dependencias de bibliotecas para el acceso a los sensores del Sistema Empotrado.

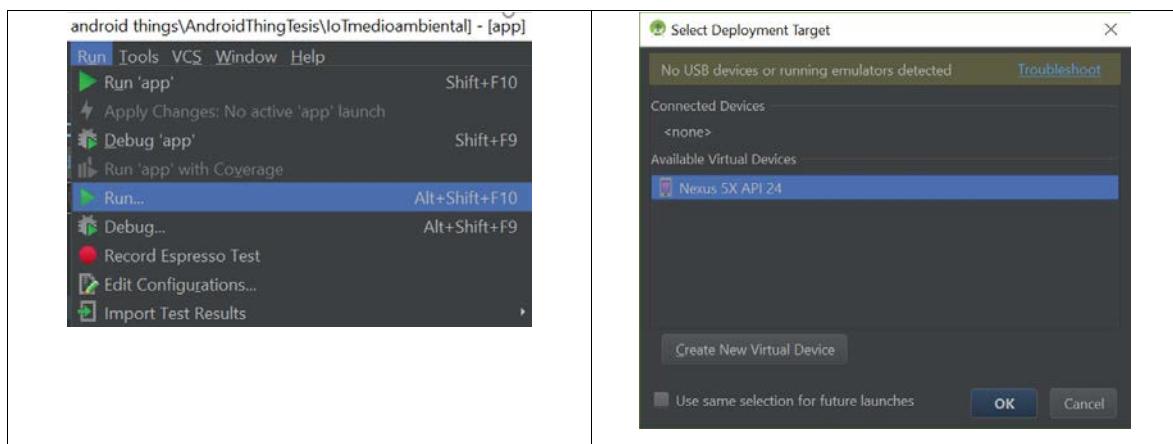
```
1. apply plugin: 'com.android.application'
2.
3. android {
4.     compileSdkVersion 25
5.     buildToolsVersion '25.0.2'
6.
7.     defaultConfig {
8.         applicationId "com.example.androidthings.weatherstation"
9.         minSdkVersion 24
10.        targetSdkVersion 25
11.        versionCode 1
12.        versionName "1.0"
13.    }
14.    buildTypes {
15.        debug {
16.            buildConfigField "String", "PROJECT_ID", '"androidthingsvic-
170713"'
17.            buildConfigField "String", "PUBSUB_TOPIC", '"temaandroidthings"'
18.        }
19.    }
20. }
21.
22. dependencies {
23.     provided 'com.google.android.things:androidthings:0.4-devpreview'
24.
25.     compile 'com.google.android.things.contrib:driver-button:0.3'
26.     compile 'com.google.android.things.contrib:driver-bmx280:0.2'
27.     compile 'com.google.android.things.contrib:driver-ht16k33:0.3'
28.     compile 'com.google.android.things.contrib:driver-apa102:0.3'
29.     compile 'com.google.android.things.contrib:driver-pwmspeaker:0.2'
30.     compile('com.google.api-client:google-api-client-android:1.22.0') {
31.         exclude group: 'org.apache.httpcomponents'
32.     }
33.     compile('com.google.apis:google-api-services-pubsub:v1-rev12-1.22.0') {
34.         exclude group: 'org.apache.httpcomponents'
35.     }
36. }
```

Código 17: Builde.gradle.

7.5.5 Despliegue de la aplicación Android Things

El despliegue de la aplicación Android Things consiste en transferir la aplicación (con extensión apk) al Sistema Empotrado. Para realizar esta tarea, fue necesario configurar previamente la conexión entre la computadora y el Sistema Empotrado tal como se realizó en la sección 7.4.1.3. Una vez configurada la conexión, el despliegue se efectúo siguiendo los pasos de la Tabla 20.

Tabla 20: Pasos para desplegar la aplicación.



7.6 Desarrollo de la Capa de Computación en la nube

La Capa de Computación en la nube es la que permite capturar, procesar, almacenar y analizar la información recolectada en la Capa de Detección, para llevar a cabo el proceso de computación; al igual que en la Figura 32, se utilizaron los servicios: Cloud Pub/Sub, Cloud DataFlow, BigQuery, Cloud Machine Learning, Compute Engine y Firebase Cloud Messaging.

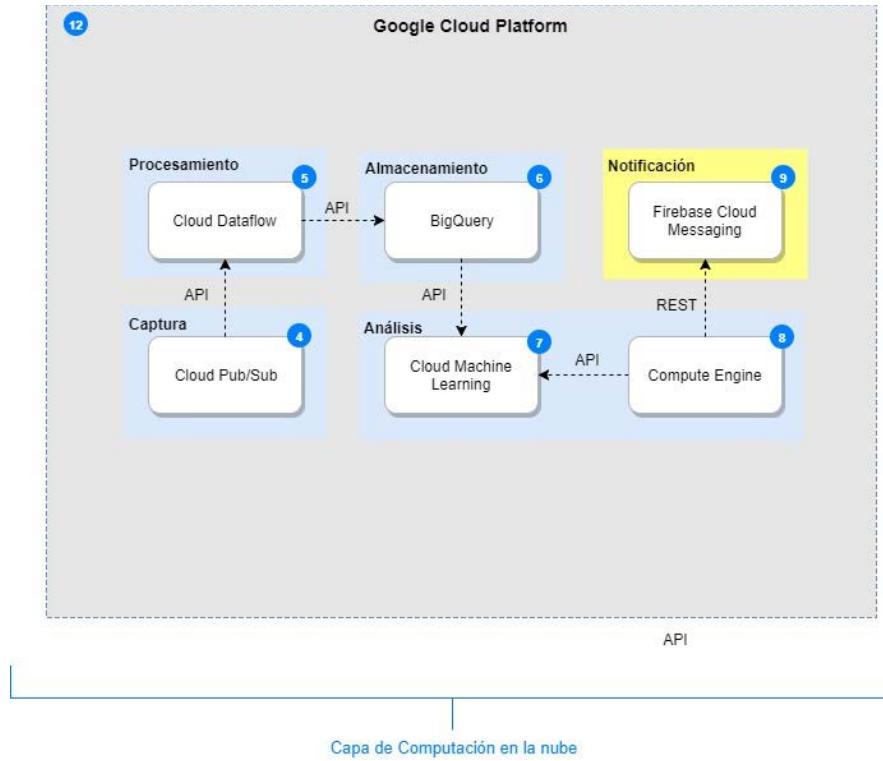


Figura 32: Capa de Computación en la nube.

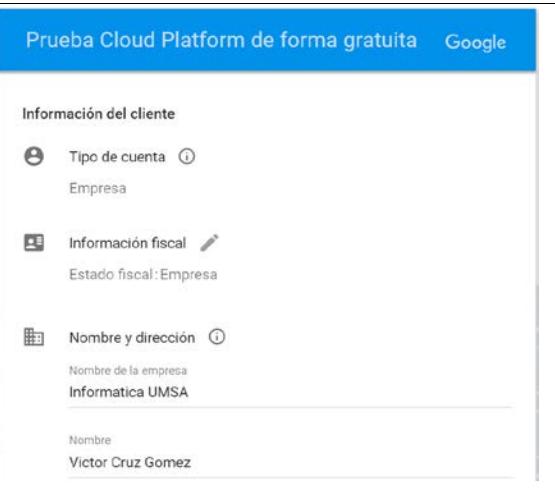
7.6.1 Implementación de Google Cloud Platform

El establecimiento de los servicios de la Capa de Computación, se efectúo en la plataforma de computación Google Cloud Platform. El único requisito necesario para la implementación de los servicios en la nube es poseer una cuenta de usuario en Gmail activada para acceder a la Consola de Administración de Google Cloud Platform.

7.6.2 Activación de la cuenta de servicio

La activación de la cuenta de servicio consiste en habilitar una cuenta de Gmail para tener acceso a la Consola de Administración de Google Cloud Platform. Al activar la cuenta de servicio, Google otorga un crédito de USD 300 para explorar Google Cloud Platform durante 12 meses. La activación de la cuenta se realiza por medio de la dirección [52] donde se requiere llenar la información solicitada en la Tabla 21.

Tabla 21: Pasos para la activación de la cuenta de servicio.

	 <p>Google Cloud Platform</p> <p>Te damos la bienvenida, maria del carmen.</p> <p>Crea y administra tus instancias, discos, redes y otros recursos de Google Cloud Platform desde un solo lugar.</p> <p>Email preferences and Terms of Service</p> <p>Quiero recibir por correo electrónico información sobre actualizaciones de funciones del producto, sugerencias de rendimiento, encuestas de opinión y ofertas especiales.</p> <p><input checked="" type="radio"/> Sí <input type="radio"/> No</p> <p>Acepto que mi uso de cualquier servicio y las API relacionadas esté sujeto a mi cumplimiento de las Condiciones del servicio aplicables.</p> <p><input checked="" type="radio"/> Sí <input type="radio"/> No</p> <p>ACEPTAR Y CONTINUAR</p>
 <p>Prueba Cloud Platform de forma gratuita Google</p> <p>País</p> <p>España</p> <p>Aceptaciones</p> <p>Quiero recibir por correo electrónico información sobre actualizaciones de funciones del producto, sugerencias de rendimiento, encuestas de opinión y ofertas especiales.</p> <p><input checked="" type="radio"/> Sí <input type="radio"/> No</p> <p>Leí y acepto las Condiciones del servicio de la prueba gratuita de Google Cloud Platform. Es necesario para continuar.</p> <p><input checked="" type="radio"/> Sí <input type="radio"/> No</p> <p>Aceptar y continuar</p>	 <p>Prueba Cloud Platform de forma gratuita Google</p> <p>Información del cliente</p> <p>Tipo de cuenta (i)</p> <p>Empresa</p> <p>Información fiscal (i)</p> <p>Estado fiscal: Empresa</p> <p>Nombre y dirección (i)</p> <p>Nombre de la empresa Informatica UMSA</p> <p>Nombre Victor Cruz Gomez</p>

7.6.3 Creación del proyecto Google Cloud Platform

El primer paso consiste en la creación del proyecto Google Cloud Platform. El proyecto permitirá administrar todos los servicios planteados en la Capa de Detección.

La creación del proyecto se inicia en la opción Panel de Control donde se elige la opción “Crear” y posteriormente se establece un nombre tal y como se observa en la Tabla 22.

Tabla 22: Pasos para la creación de un proyecto en Google Cloud Platform.

7.6.4 Servicio Cloud Pub/Sub

El servicio de Cloud Pub/Sub de la Figura 33, es un servicio global de mensajería fiable en tiempo real. Permite enviar mensajes entre aplicaciones y servicios. Implementa el patrón Publicación y Suscripción para el envío de mensajes y está diseñado para proporcionar mensajes con varios remitentes y destinatarios. Este servicio se encarga de capturar todos los datos generados por los Sistemas Empotrados y enviarlos al servicio Cloud Dataflow para su procesamiento.



Figura 33: Servicio Cloud Pub/Sub.

Arquitectura del Servicio Cloud Pub/Sub

Dentro de este servicio se crearon dos temas tal como se observa en la Figura 34. Se creó un tema llamado “Tema Sensor” para capturar los mensajes generados por los Sistemas Empotrados y otro tema llamado “Tema Cuestionario” para capturar los resultados de la evaluación aplicados a los usuarios del sistema. También en la Figura 53 se observan las suscripciones creadas: “Suscripción DataFlow Sensor” y “Suscripción DataFlow Cuestionario”. Las suscripciones permiten al servicio DataFlow extraer, transformar y cargar los mensajes hacia el servicio de almacenamiento BigQuery.

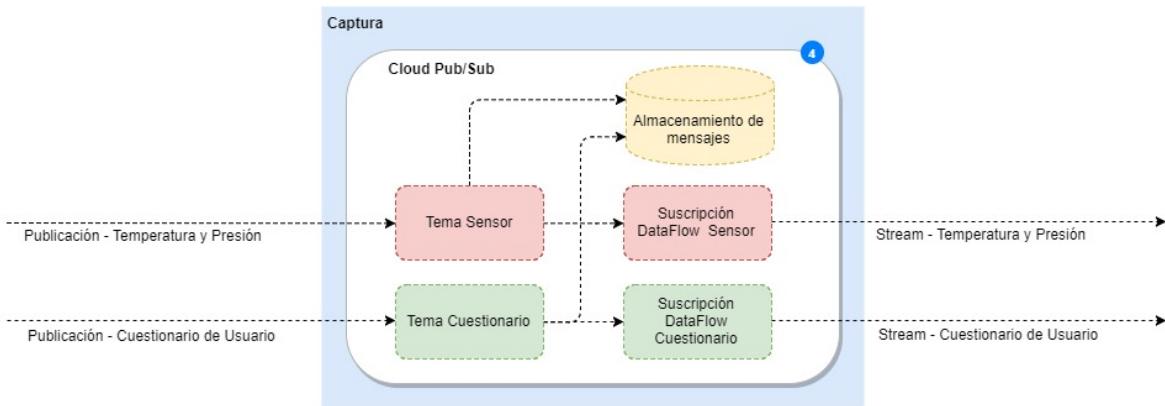


Figura 34: Servicio Cloud Pub/Sub detallado.

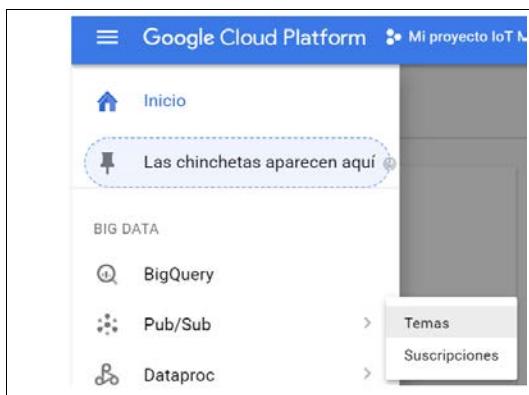
Configuración del Servicio Cloud Pub/Sub

La configuración del servicio Cloud Pub/Sub consistió en:

- Habilitar la API Pub/Sub
- Crear el un tema de publicación.
- Crear un suscriptor
- Otorgar permisos de publicación

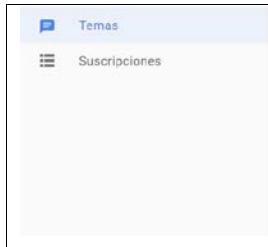
Los pasos para habilitar la API Pub/Sub y la creación de los temas de publicación, se detallan en la Tabla 23.

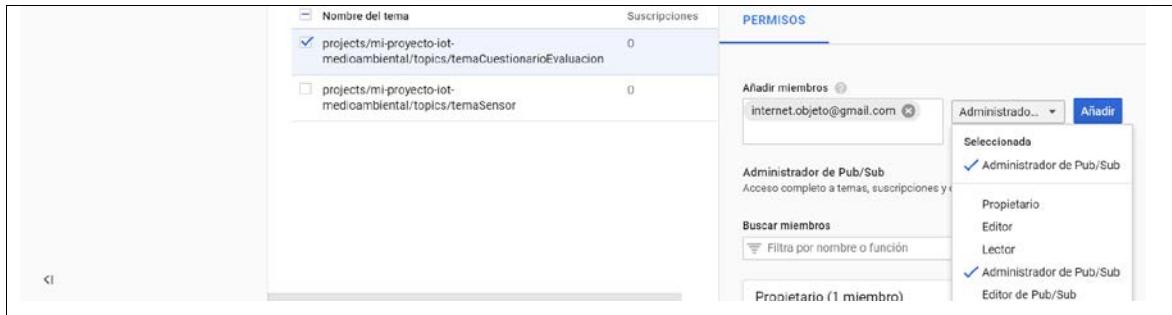
Tabla 23: Pasos para habilitar la API Pub/Sub y creación de temas.

	<p>Big Data Pub/Sub</p> <p>Mensajería en tiempo real fiable Conecta tus servicios a un servicio de mensajería fiable, asíncrono, que admite varios remitentes y destinatarios, y que está alojado en la infraestructura de Google. Para empezar, crea un tema para publicar mensajes asíncronos a varios suscriptores. Más información</p> <p>Habilitar API</p>																								
<p>Big Data Pub/Sub</p> <p>Mensajería en tiempo real fiable Conecta tus servicios a un servicio de mensajería fiable, asíncrono, que admite varios remitentes y destinatarios, y que está alojado en la infraestructura de Google. Para empezar, crea un tema para publicar mensajes asíncronos a varios suscriptores. Más información</p> <p>Crear un tema</p>	<p>Crear un tema</p> <p>Los temas reenvían mensajes de editores a suscriptores.</p> <p>Nombre <input type="text" value="projects/mi-proyecto-iot-medioambiental/topics/ temaSensor"/></p> <p>CANCELAR CREAR</p>																								
	<p>Google Cloud Platform Mi proyecto IoT Medioam...</p> <table border="1"> <thead> <tr> <th>Temas</th> <th>CREAR TEMA</th> <th>ELIMINAR</th> <th>MOSTRAR PANEL DE INFORMACIÓN</th> </tr> </thead> <tbody> <tr> <td>Temas</td> <td><input type="button" value="Filtrar por nombre de tema"/></td> <td></td> <td></td> </tr> <tr> <td>Suscripciones</td> <td></td> <td></td> <td></td> </tr> <tr> <td></td> <td><input type="checkbox"/> Nombre del tema</td> <td>Suscripciones</td> <td></td> </tr> <tr> <td></td> <td><input type="checkbox"/> projects/mi-proyecto-iot-medioambiental/topics/temaCuestionarioEvaluacion</td> <td>0</td> <td><input type="button" value="::"/></td> </tr> <tr> <td></td> <td><input type="checkbox"/> projects/mi-proyecto-iot-medioambiental/topics/temaSensor</td> <td>0</td> <td><input type="button" value="::"/></td> </tr> </tbody> </table>	Temas	CREAR TEMA	ELIMINAR	MOSTRAR PANEL DE INFORMACIÓN	Temas	<input type="button" value="Filtrar por nombre de tema"/>			Suscripciones					<input type="checkbox"/> Nombre del tema	Suscripciones			<input type="checkbox"/> projects/mi-proyecto-iot-medioambiental/topics/temaCuestionarioEvaluacion	0	<input type="button" value="::"/>		<input type="checkbox"/> projects/mi-proyecto-iot-medioambiental/topics/temaSensor	0	<input type="button" value="::"/>
Temas	CREAR TEMA	ELIMINAR	MOSTRAR PANEL DE INFORMACIÓN																						
Temas	<input type="button" value="Filtrar por nombre de tema"/>																								
Suscripciones																									
	<input type="checkbox"/> Nombre del tema	Suscripciones																							
	<input type="checkbox"/> projects/mi-proyecto-iot-medioambiental/topics/temaCuestionarioEvaluacion	0	<input type="button" value="::"/>																						
	<input type="checkbox"/> projects/mi-proyecto-iot-medioambiental/topics/temaSensor	0	<input type="button" value="::"/>																						

Los pasos para la otorgación de permisos de publicación a la cuenta de Gmail, se detallan en la Tabla 24.

Tabla 24: Pasos para otorgar permisos de publicación.

	<p>Filtrar por nombre de tema</p> <table border="1"> <thead> <tr> <th>Nombre del tema</th> <th>Suscripciones</th> <th>⋮</th> </tr> </thead> <tbody> <tr> <td>projects/mi-proyecto-iot-medioambiental/topics/temaCuestionarioEvaluacion</td> <td>0</td> <td><input type="button" value="::"/></td> </tr> <tr> <td>projects/mi-proyecto-iot-medioambiental/topics/temaSensor</td> <td>0</td> <td><input type="button" value="::"/></td> </tr> </tbody> </table> <div data-bbox="1117 1679 1264 1784" style="background-color: #e0e0e0; padding: 5px;"> <input type="button" value="Suscripción nueva"/> <input type="button" value="Publicar mensaje"/> <input type="button" value="Eliminar"/> <input type="button" value="Permisos"/> </div>	Nombre del tema	Suscripciones	⋮	projects/mi-proyecto-iot-medioambiental/topics/temaCuestionarioEvaluacion	0	<input type="button" value="::"/>	projects/mi-proyecto-iot-medioambiental/topics/temaSensor	0	<input type="button" value="::"/>
Nombre del tema	Suscripciones	⋮								
projects/mi-proyecto-iot-medioambiental/topics/temaCuestionarioEvaluacion	0	<input type="button" value="::"/>								
projects/mi-proyecto-iot-medioambiental/topics/temaSensor	0	<input type="button" value="::"/>								



7.6.5 Servicio Cloud Dataflow

Cloud Dataflow de la Figura 34, es un servicio para el procesamiento de datos por lotes y transmisión en tiempo real. En este servicio las tareas de procesamiento de datos se representan mediante canalizaciones. Una canalización lee los datos de entrada, los transforma y a continuación produce datos de salida. Entre las transformaciones que realizan las canalizaciones se incluyen el filtrado, la agrupación, la comparación y la unificación de datos.



Figura 35: Servicio Cloud Dataflow.

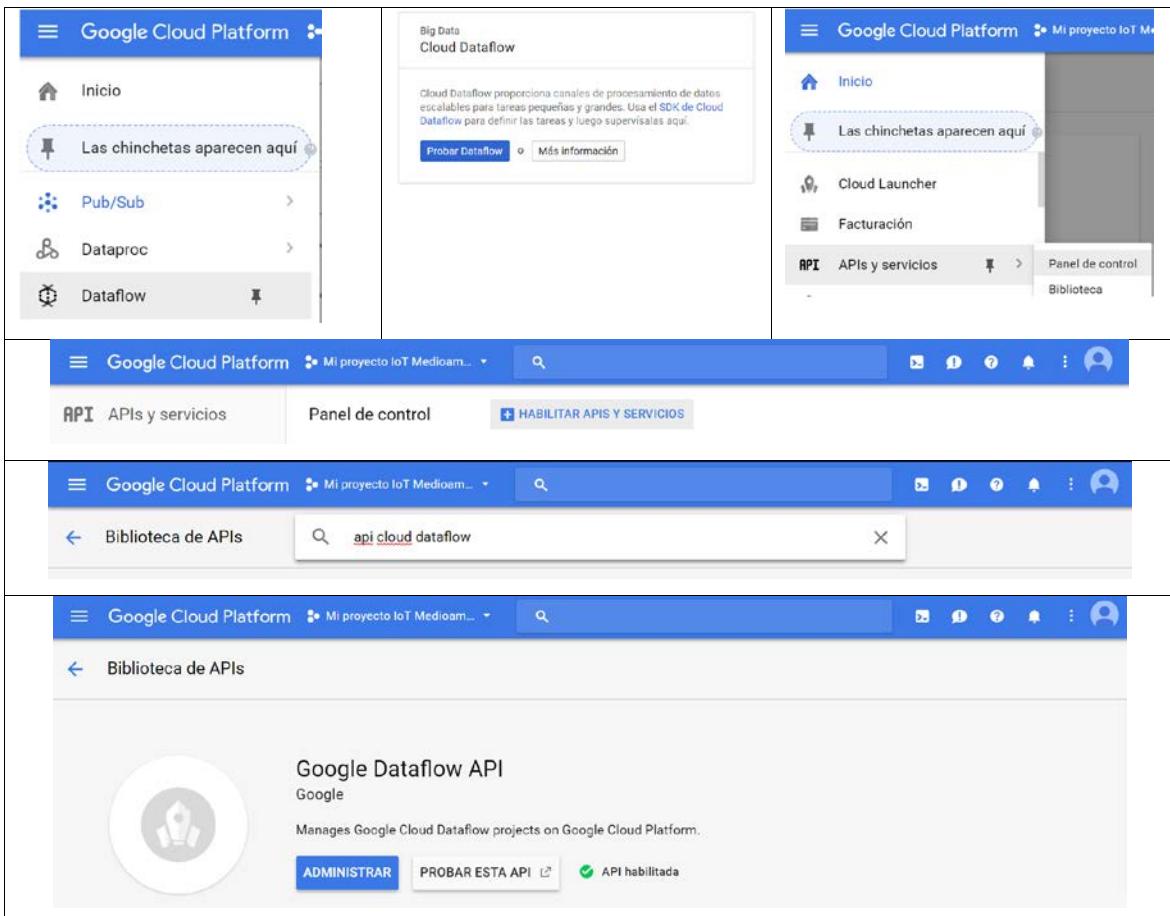
Para el procesamiento de los datos, Cloud Dataflow hace uso de Beam [63], una biblioteca para el análisis de cadenas de datos masivos. La biblioteca Beam funciona con tres conceptos básicos:

- Pipeline o Canalización, es una serie de transformaciones por la que un conjunto de datos debe pasar.
- PCollection, es la colección de datos.
- PTransform, es un tipo de transformación que es aplicada a colecciones de datos.

Configuración del Servicio Cloud Dataflow

Los pasos para configurar la API Cloud Dataflow se detallan en la Tabla 25.

Tabla 25: Pasos para habilitar la API Cloud Dataflow.



Crear una aplicación de procesamiento de cadenas

La aplicación de procesamiento de cadenas es un programa escrito en Java que utiliza bibliotecas Beam para el análisis de cadenas de datos masivos. La aplicación permite transformar los mensajes procedentes del servicio Cloud Pub/Sub y obtener un resultado con mensajes formateados tal como se observa en la Figura 36. Para realizar el procesamiento de las cadenas de mensajes la aplicación efectuara las siguientes acciones:

- Creará un pipeline o canalización entre el servicio Cloud Pub/Sub y el Cloud Dataflow.
- Una vez creado la canalización, el programa procederá a leer los mensajes del servicio Cloud Pub/Sub y los almacenará temporalmente en una colección de datos PCollection. El PCollection puede ser limitada, lo que significa que proviene de una fuente fija como un archivo o ilimitada. En este caso la colección de datos será ilimitada ya que la fuente de datos es un servicio Cloud Pub/Sub
- Posteriormente, se aplicarán diferentes transformaciones a los mensajes por medio de PTransform, como por ejemplo agregar a cada mensaje la fecha y hora del servidor.
- Finalmente, una nueva canalización con los mensajes formateados.

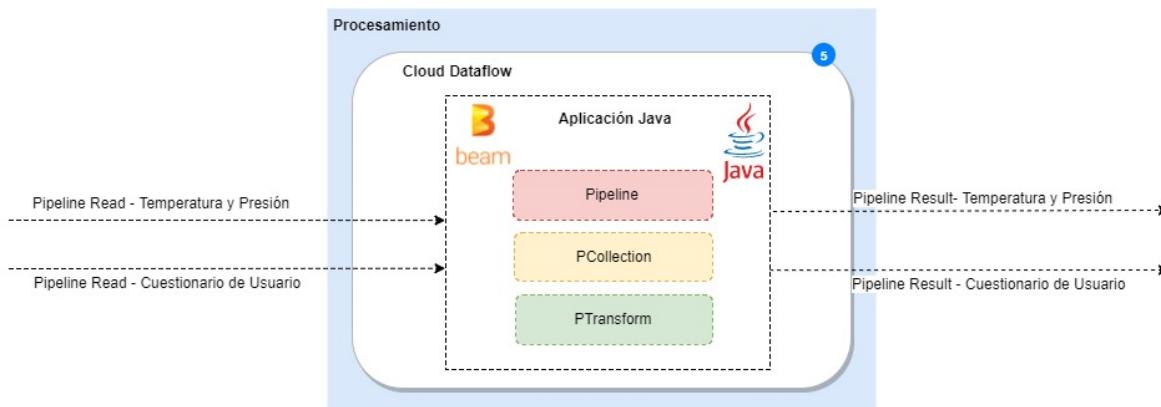


Figura 36: Arquitectura interna del servicio Cloud Dataflow.

Implementación de la aplicación de procesamiento de cadenas

Tal como se comentó anteriormente, la aplicación para el procesamiento de cadenas será escrita en el lenguaje de programación Java utilizando el IDE Eclipse. En este caso se utilizó el IDE Eclipse por la facilidad de integración con Google Cloud Platform ya que el mismo incluye un plugin para ingresar las credenciales de conexión con la nube.

Requerimiento de Software

Los requerimientos de Software para desarrollar la aplicación de procesamiento de cadena de datos en Cloud DataFlow fueron los siguientes:

- Eclipse Neon 2 Release 4.6.2
- Google Cloud Tools para Eclipse 1.4.1 (Plugin)
- Bibliotecas en Eclipse Neon 2 Release 4.6.2 (Maven)
 - jackson-core 2.9.1
 - google-http-client 1.22.0
 - google-api-client 1.22.0
 - google-cloud-dataflow-java-sdk-parent 1.9.1
 - google-api-services-dataflow v1b3-rev43-1.22.0
 - google-api-services-bigquery v2-rev295-1.22.0

Estructura del proyecto

El Proyecto implementado en el IDE Eclipse tiene estructura de la Figura 37.

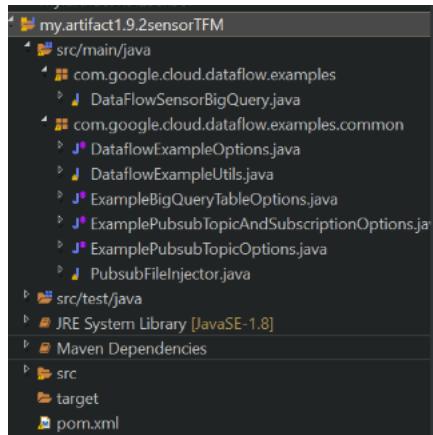


Figura 37: Estructura del proyecto Dataflow.

Del Proyecto las clases “DataFlowSensorBigQuery.java” y “DataFlowCuestionarioBigQuery.java” son las principales. Las demás clases son utilizadas para manejar eventos de los servicios Cloud Pub/Sub y Cloud BigQuery.

DataFlowSensorBigQuery.java

La clase “DataFlowSensorBigQuery.java” y “DataFlowCuestionarioBigQuery.java” son similares, ambas permiten realizar el procesamiento de los mensajes procedentes del Cloud Pub/Sub para luego almacenarlo en sus respectivas tablas del servicio BigQuery.

En el fragmento de Código 18, en la línea 3, se observa una definición WINDOW_SIZE que se refiere al tamaño de la “Ventana”; una “Ventana” en Cloud Dataflow se refiere a la división de los elementos de una PCollection según sus marcas de tiempo [63] . El método “getEsquemaTabla” define la estructura de la tabla donde se almacenarán los mensajes del servicio Cloud Pub/Sub.

```
1. public class DataFlowSensorBigQuery {  
2.     private static final Logger LOG = LoggerFactory.getLogger(DataFlowSensorBigQu  
ery.class);  
3.     static final int WINDOW_SIZE = 0;  
4.  
5.     private static TableSchema getEsquemaTabla() {  
6.         List<TableFieldSchema> fields = new ArrayList<>();  
7.         fields.add(new TableFieldSchema(). setName("deviceid"). setType("STRING"));  
8.         fields.add(new TableFieldSchema(). setName("channel"). setType("STRING"));  
9.         fields.add(new TableFieldSchema(). setName("timestamp"). setType("STRING"));  
10.        fields.add(new TableFieldSchema(). setName("temperature"). setType("STRING"))  
11.        fields.add(new TableFieldSchema(). setName("pressure"). setType("STRING"));  
12.        fields.add(new TableFieldSchema(). setName("datetime"). setType("STRING"));  
13.        TableSchema schema = new TableSchema(). setFields(fields);  
14.        return schema;  
15.    }  
Código 18: DataFlowSensorBigQuery.java.
```

En el fragmento de Código 19, se establece la configuración de la tabla que se creará en el servicio BigQuery.

```
1. private static TableReference getReferenciaTabla(IOpciones options) {  
2.     TableReference tableRef = new TableReference();  
3.     tableRef.setProjectId(options.getProject());  
4.     tableRef.setDatasetId(options.getBigQueryDataset());  
5.     tableRef.setTableId(options.getBigQueryTable());  
6.     return tableRef;  
7. }  
Código 19: DataFlowSensorBigQuery.java - Preferencias de tabla.
```

En el fragmento de Código 20, se establece la configuración de ejecución del programa destacando elementos tales como: el tamaño de la “Ventana” y el tipo de fuente de datos (limitado o ilimitado).

```
1. public interface IOpciones extends DataflowExampleOptions, ExamplePubsubTopicOpti  
ons, ExampleBigQueryTableOptions {  
2.     @Description("Duración de la ventana fija en minutos")  
3.     @Default.Integer(WINDOW_SIZE)
```

```

4.     Integer getWindowSize();
5.     void setWindowSize(Integer value);
6.     @Description("Si hay que ejecutar el pipeline con entrada de datos ilimitado
7.     boolean isUnbounded();
8.     void setUnbounded(boolean value);
9. }

```

Código 20: DataFlowSensorBigQuery.java - Configuración de ejecución del programa.

En el fragmento de Código 21, se ha creado una clase, y dentro de este un método para extraer los datos de los mensajes (que se encuentran en formato json). En la línea 1, los dos parámetros de la clase genérica “DoFn” indican parámetros de entrada y salida de datos. Es decir, el parámetro “String” se corresponde con los mensajes obtenidos en formato Json desde el Cloud Pub/Sub como el de la Figura 38. El otro parámetro “TableRow” se corresponde con el resultado obtenido en la extracción de datos. El método “processElement” tiene un parámetro “ProcessContext” que hace referencia a un elemento de la colección de datos PCollection. Para cada elemento del PCollection se obtiene un objeto de tipo Sensor por medio del método “convertirJsonObjeto”. Por último, el objeto sensor, que contiene los datos del mensaje, se utiliza para obtener un determinado valor y transformarlo en un campo del tipo TableRow.

```

1. static class ExtraerDatosSensorFn extends DoFn<String, TableRow> {
2.     @Override
3.     public void processElement(ProcessContext c) {
4.         Sensor sensor = convertirJsonObjeto(c.element());
5.         TableRow row = new TableRow();
6.         set("deviceId", sensor.getDeviceId());
7.         set("channel", sensor.getChannel());
8.         set("timestamp", sensor.getTimestamp());
9.         set("temperature", sensor.getData().getTemperature());
10.        set("pressure", sensor.getData().getPressure());
11.        set("datetime", new DateTime(sensor.getTimestamp()).toString());
12.        c.output(row);
13.    }
14. }

```

Código 21: DataFlowSensorBigQuery.java - Extracción de datos.

The screenshot shows a terminal window titled "Google Cloud SDK Shell". The command entered is "C:\Program Files (x86)\Google\Cloud SDK>gcloud beta pubsub subscriptions pull --auto-ack suscripcionprueba". Below the command, a table displays the pulled message. The table has three columns: DATA, MESSAGE_ID, and ATTRIBUTES. The DATA column contains the JSON message: {"deviceId": "rp13", "channel": "pubsub", "timestamp": 1505416224536, "data": {"temperature": "41.963936", "pressure": "941.65216"}}. The MESSAGE_ID column contains the ID: 147304388858468. The ATTRIBUTES column is empty.

DATA	MESSAGE_ID	ATTRIBUTES
{"deviceId": "rp13", "channel": "pubsub", "timestamp": 1505416224536, "data": {"temperature": "41.963936", "pressure": "941.65216"}}	147304388858468	

Figura 38: Objeto JSON publicado en el Cloud Pub/Sub.

En el fragmento del Código 22 se observa el método principal del programa, es el lugar donde se instancian los objetos para el procesamiento de la cadena de datos. En la línea 2 se establecieron los parámetros de configuración para crear una tabla en el servicio BigQuery. En la línea 4 se define el esquema de la tabla que se creará en el servicio BigQuery. En la línea 5 se establecieron los parámetros del Dataflow, en nuestro caso se establecerá una canalización de datos ilimitada. En las líneas 7 y 8 se procede a leer los datos del servicio Cloud Pub/Sub y almacenarlos temporalmente en una colección de datos PCollection. Finalmente, en las líneas 10 al 16 se aplican transformaciones a la PCollection y el resultado se escribe en una tabla del servicio BigQuery.

```

1. public static void main(String[] args) throws IOException {
2.     IOpciones opciones = PipelineOptionsFactory.fromArgs(args)
3.         .withValidation().as(IOpciones.class);
4.     opciones.setBigQuerySchema(getEsquemaTabla());
5.     DataflowExampleUtils exampleDataflowUtils = new
6.         DataflowExampleUtils(opciones, opciones.isUnbounded());
7.     Pipeline pipeline = Pipeline.create(opciones);
8.     PCollection<String> input= pipeline
9.         .apply(PubsubIO.Read.topic(opciones.getPubsubTopic()));
10.    input.apply(ParDo.of(new ExtraerDatosSensorFn()))
11.        .apply(BigQueryIO.Write.to(getReferenciaTabla(opciones))
12.            .withSchema(getEsquemaTabla())
13.                .withCreateDisposition(BigQueryIO.Write.CreateDisposition
14.                    .CREATE_IF_NEEDED)
15.                .withWriteDisposition(BigQueryIO.Write.WriteDisposition
16.                    .WRITE_APPEND));
17.    PipelineResult result = pipeline.run();
18. }
```

Código 22: DataFlowSensorBigQuery.java - Método principal del programa.

7.6.6 Servicio BigQuery

BigQuery de la Figura 39, es un servicio de análisis y almacenamiento de datos a gran escala. Permite realizar el análisis de grandes conjuntos de datos a través de consultas SQL (Estándar 2011) súper rápidas utilizando el poder de procesamiento de la infraestructura de Google.

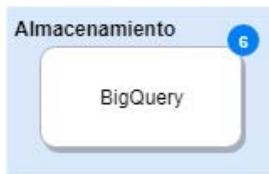


Figura 39: Servicio BigQuery.

Arquitectura del Servicio BigQuery

La arquitectura interna del Servicio BigQuery es la que se observa en la Figura 40. En ella se puede ver una base de datos con dos tablas. La tabla “tblSensor” es la estructura de datos donde se almacena la información recolectada por los Sistemas Empotrados. La tabla “tblCuestionario” es la estructura de datos donde se almacenan los resultados de los cuestionarios aplicados a los usuarios del Sistema. Las tablas de BigQuery pueden ser creadas desde la Consola de Administración o desde el código Java. En nuestro caso las tablas de BigQuery se crearon desde el servicio Cloud Dataflow por medio de la aplicación implementada para el procesamiento de los mensajes procedentes del servicio Cloud Pub/Sub.

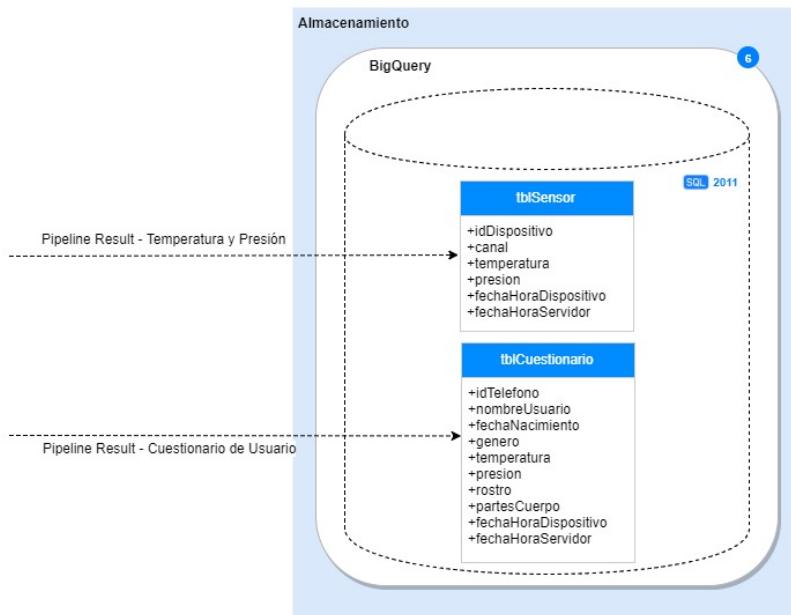


Figura 40: Arquitectura interna del Servicio BigQuery.

Configuración del Servicio Cloud Dataflow

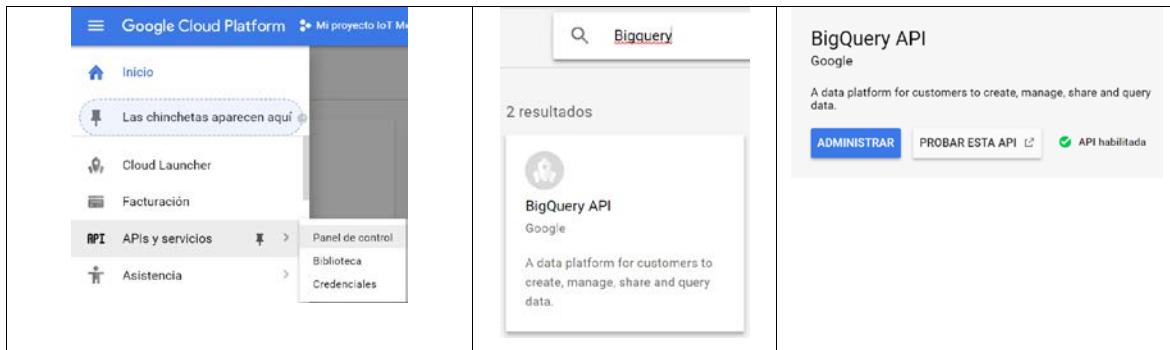
Para utilizar este servicio, antes se tiene que habilitar el mismo desde la Consola de Administración y posteriormente crear las tablas de almacenamiento de datos; para ello se tiene que seguir los siguientes pasos:

- Habilitar la API de BigQuery
- Otorgar permisos a la cuenta de Google
- Crear las tablas de almacenamiento

Habilitación de la API de BigQuery

La habilitación de la API de BigQuery se realiza desde la Consola de Administración de Google Cloud Platform. Para habilitar la API de BigQuery se seguirán los pasos de la Tabla 26.

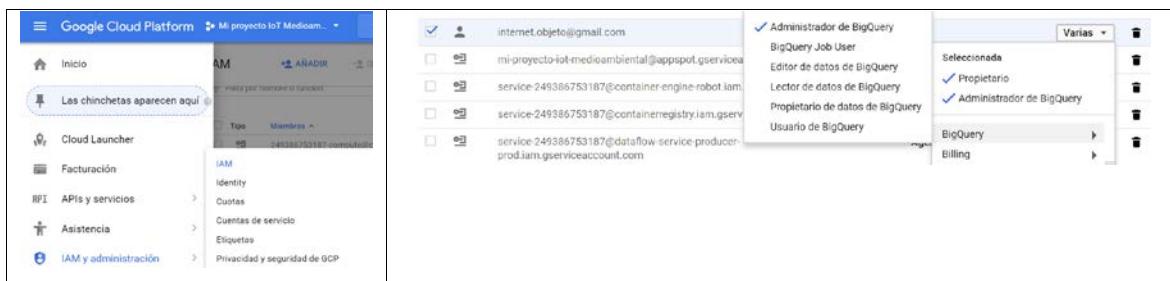
Tabla 26: Pasos para habilitar la API BigQuery.



Otorgar permisos a la cuenta de Google

Después de habilitar la API de BigQuery, el siguiente paso es otorgar permisos a la cuenta de Google para la administración de la base de datos. Para ello, deben seguirse las instrucciones de la Tabla 27.

Tabla 27: Pasos para otorgar permisos a la cuenta de Google.



7.6.7 Servicio Compute Engine

Compute Engine de la Figura 41, es un servicio para la administración y ejecución de máquinas virtuales en la infraestructura de Google.

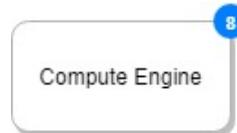


Figura 41: Servicio Compute Engine.

En nuestro caso se utilizó el servicio Compute Engine para obtener los resultados de la predicción almacenados en el servicio Cloud Machine Learning para luego enviarlos en forma de notificación al usuario del sistema. El servicio Compute Engine incluye una máquina virtual “Debían 8” con un Servicio Web basado en REST tal como se observa en la Figura 42. El Servicio Web basado en REST, es el responsable de:

- Obtener los resultados de la predicción del servicio Cloud Machine Learning.
- Enviar una petición POST al ser servidor de notificación Firebase Cloud Messaging.

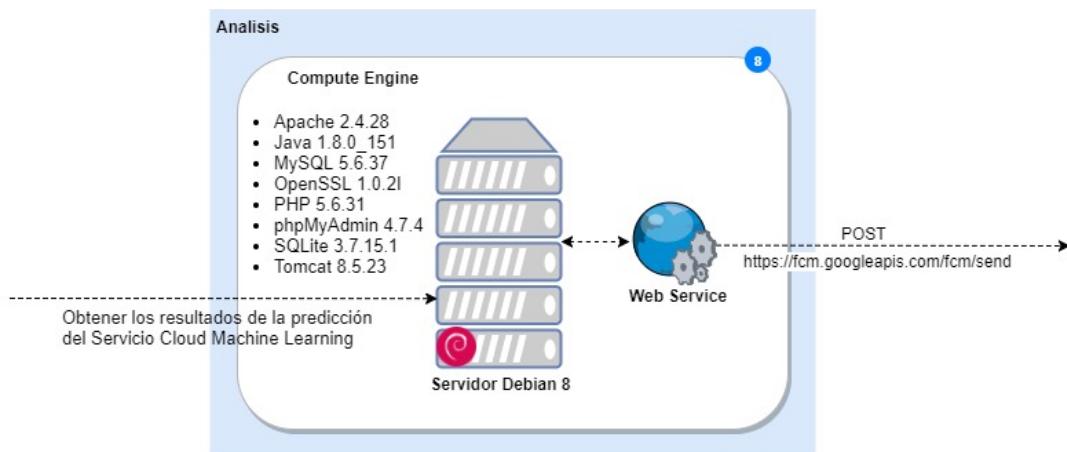


Figura 42: Servicio Compute Engine detallado.

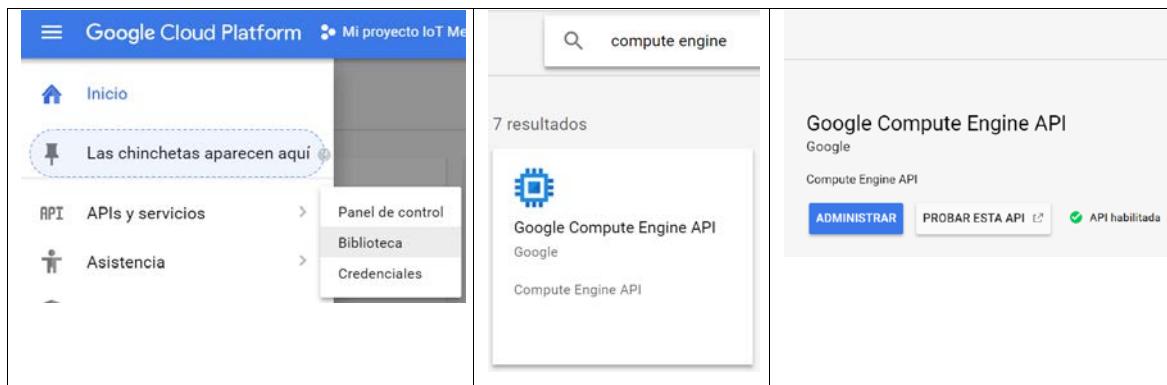
Para utilizar el servicio de Compute Engine, antes se tiene que habilitar el mismo desde la Consola de Administración y posteriormente crear una máquina virtual, para ello se efectuaron los siguientes pasos:

- Habilitar Google Compute Engine API.
- Otorgar permisos a la cuenta de Google
- Crear la máquina virtual

Habilitar Google Compute Engine API

La habilitación de la API de Compute Engine se realizó desde la Consola de Administración de Google Cloud Platform, tal como se observa en la Tabla 28.

Tabla 28: Pasos para habilitar la API Compute Engine.



Otorgar permisos a la cuenta de Google

El siguiente paso, después de habilitar la API de Google Compute Engine, consiste en otorgar permisos a la cuenta de Google. Las instrucciones necesarias para ello se encuentran en la Tabla 29.

Tabla 29: Pasos para otorgar permisos a la cuenta de Google.

The screenshot shows the Google Cloud Platform's IAM & administration interface. On the left, there's a sidebar with options like Inicio, Cloud Launcher, Facturación, APIs y servicios, Asistencia, and IAM y administración. The main area shows a list of users and service accounts under the 'Membres' tab. A specific service account, 'mi-proyecto-iot-medioambiental@appspot.gserviceaccount.com', is selected. On the right, a detailed view of its permissions is shown, with several checkboxes checked under the 'Propietario' and 'Administrador de BigQuery' sections. A context menu is open over these checkboxes, with 'Varías' selected. The menu lists various Compute Engine roles: Administrador de Compute Storage, Administrador de instancias de Compute (beta), Administrador de instancias de Compute (v1), Administrador de red de Compute, Administrador de seguridad de Compute, Compute Admin, and Compute Load Balancer Admin. A tooltip indicates that selecting these roles grants 'Control total sobre todos los recursos de Compute Engine.'

Crear la máquina virtual

Finalmente, para poder alojar el servicio web basado en REST se creará una máquina virtual. La máquina virtual puede ser cualquiera de las imágenes ofrecidas como disco de arranque. Es decir, al momento de crear Google Cloud Platform, ofrece un conjunto de imágenes con los sistemas operativo más conocidos. Por ejemplo, versiones Windows Server, Linux u otros que se encuentran disponibles en la lista de imágenes de Google. En nuestro caso se creará una máquina virtual a partir de una distribución Linux, específicamente Debían8 y un contenedor de servicios web Tomcat. Para la creación de esta máquina virtual, se seguirán los pasos de la Tabla 30.

Tabla 30: Pasos para crear la máquina virtual Debían 8.

The screenshot shows the Google Cloud Platform's Compute Engine Instances page and a modal dialog for creating a new instance. The main page has tabs for Compute Engine, Instancias de VM, and Grupos de instancias. A 'Crear instancia' button is visible. The modal dialog is titled 'Crear una instancia'. It contains several sections: 'Disco de arranque' (with a dropdown showing 'Nuevo disco persistente estándar de 10 GB' and 'Imagen' set to 'Debian GNU/Linux 8 (jessie)'), 'Identidad y acceso de API' (with 'Cuenta de servicio' set to 'Compute Engine default service account'), and 'Alcance del acceso' (with 'Permitir el acceso predeterminado' selected). On the right, the 'Instancias de VM' section is visible, showing a summary of Compute Engine instances with their status and creation date. Below it, a 'Disco de arranque' section shows a list of available images, with 'Debian GNU/Linux 8 (jessie)' selected. Other options include 'Ubuntu 16.04 LTS (xenial)', 'Ubuntu 17.04 (zesty)', 'CentOS 7', and 'CentOS 6'. There are also tabs for 'Imágenes de la aplicación', 'Imágenes personalizadas', 'Capturas', and 'Discos disponibles'. At the bottom of the modal, there are 'Seleccionar' and 'Cancelar' buttons.

7.6.8 Configuración del servicio Firebase Cloud Messaging

Firebase Cloud Messaging(FCM) de la Figura 43, es una solución de mensajería multiplataforma que permite enviar mensajes de forma segura y gratuita hacia aplicaciones móviles y web.



Figura 43: Servicio Firebase Cloud Messaging.

El servicio de mensajería FCM, envía automáticamente mensajes a los dispositivos de los usuarios finales cuando recibe una petición de entrega. Para que el servicio FCM conozca en destino final del mensaje, previamente es necesario agregar dependencias de Firebase en la aplicación que será responsable de recibir los mensajes, que son enviados en formato JSON. Estos mensajes incluyen en su cabecera un token de identificación, para asegurar que cada mensaje sea entregado a un determinado dispositivo. El servicio FCM guarda en una base de datos los token de cada dispositivo. El token que identifica a cada dispositivo, se genera automáticamente durante la ejecución de la aplicación. Por último, el flujo de los mensajes enviados desde el servicio Compute Engine hacia los dispositivos de los usuarios finales, se observa en la Figura 44.

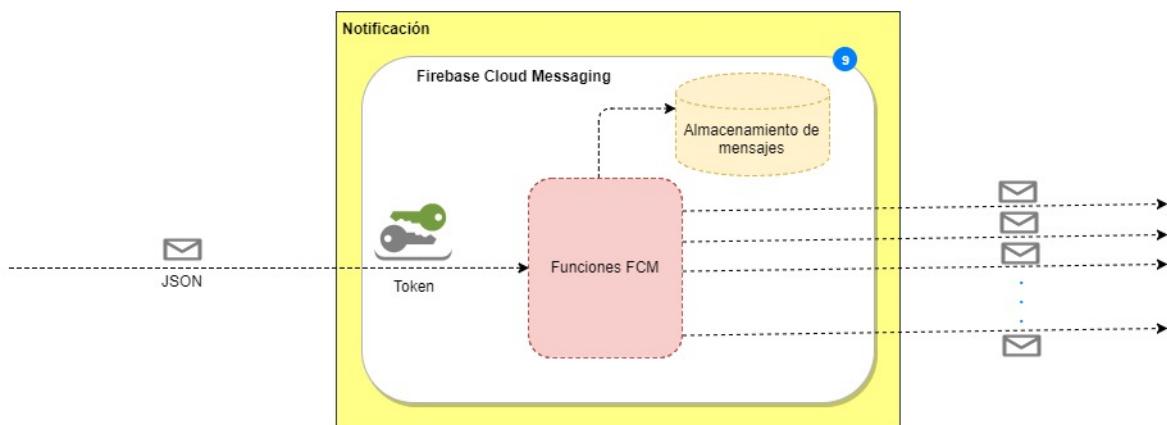


Figura 44: Servicio Firebase Cloud Messaging detallado.

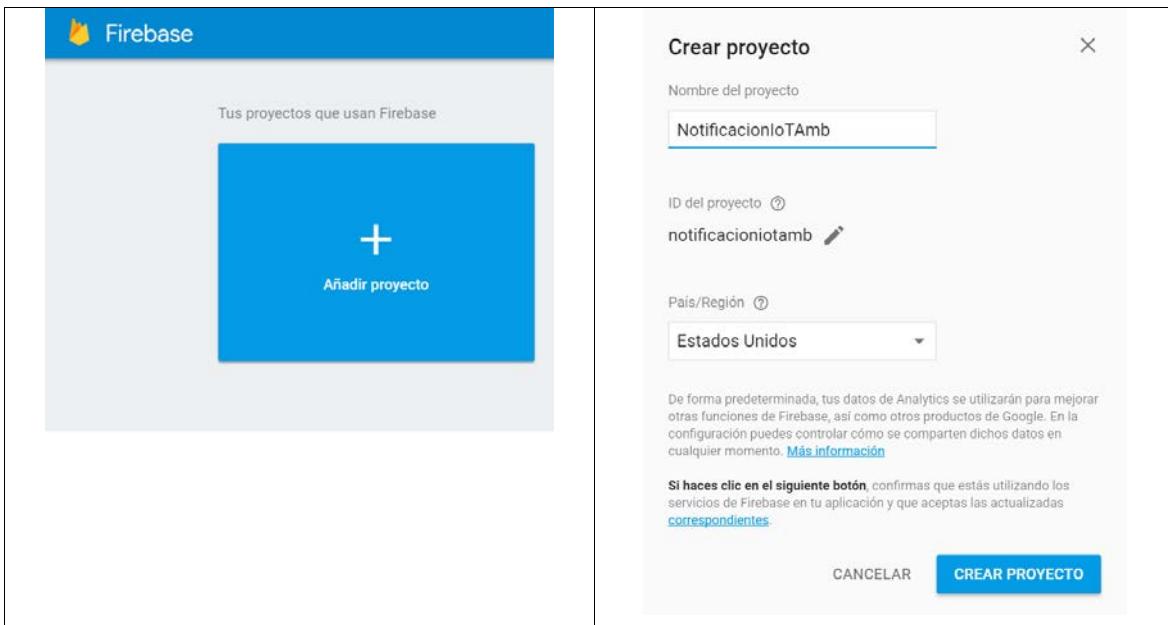
Utilizar el servicio Firebase Cloud Mensaje implica:

- Crear un proyecto en Firebase
- Obtener la clave del servidor
- Incluir el sdk en los dispositivos de usuario.

Crear un proyecto Firebase

La creación del proyecto Firebase se realiza desde la dirección [64] siguiendo los pasos de la Tabla 31.

Tabla 31: Pasos para crear un proyecto en Firebase.



Obtener la clave del servidor

Como parte de la seguridad de Firebase, la clave del servidor permite identificar a los servicios web que pretendan enviar mensajes hacia los dispositivos finales. La clave del servidor se utiliza en el header del mensaje de la solicitud POST. La obtención de la clave del servidor se obtiene siguiendo los pasos de la Tabla 32.

Tabla 32: Pasos para obtener la clave del servidor.

The screenshot shows the Firebase Project Overview interface. The top navigation bar includes the Firebase logo, the project name 'NotificacionioTAmB', a documentation link, and tabs for 'Project Overview', 'Configuración del proyecto' (which is highlighted), 'Administrar en Google Cloud Console', 'NUBE', 'ANALYTICS', 'ENLACE DE CUENTAS', and 'CUENTAS DE SERVICIO'. Below this, the 'DEVELOP' section is visible with 'Authentication' selected. The main content area is titled 'Credenciales de proyecto' and contains a table with one row. The table has two columns: 'Clave' and 'Token'. The 'Clave' column contains the value 'Clave de servidor', and the 'Token' column contains a long string of characters: 'AAAADjNI1Wk:APA91bGAI\$jkKqIApRVYyELQyxYSKcb-7EaHviu6jqDifb_PD3PEGZUhSi1aziKZz4n1Yk4M... A6ZIVaiPB8MW_cMYFu7ViZvpAxyyxHYBC76l6ZqtRBqCorkINDAOQQuA4uTZvjBAnR'. A blue button labeled 'AÑadir Clave de Servidor' is located at the bottom right of the table.

7.7 Desarrollo de la Capa de Aplicación

La Capa de Aplicación de la Figura 45, es la que se encarga de interactuar con los usuarios a través de notificaciones y cuestionarios. Esta capa es importante puesto que permitirá retroalimentar el sistema con la información generada por el usuario y así en un futuro, el sistema medioambiental por medio del servicio Cloud Machine Learning encuentre patrones de datos que afecten al estado de salud de las personas.

Para realizar esta tarea se utilizarán aplicaciones desarrolladas en Android. Las notificaciones serán entregadas por el servicio Firebase Cloud Messaging y se leerán por medio de una aplicación instalada en el teléfono móvil del usuario. Los cuestionarios se llenarán con la misma aplicación del usuario, para posteriormente enviar los datos al servicio Cloud Pub/Sub con el objetivo de que la información de los cuestionarios se almacene en una tabla de BigQuery para su posterior análisis.

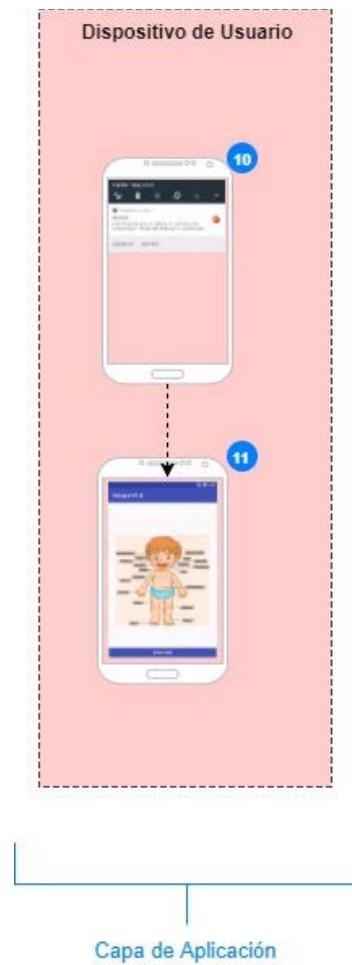


Figura 45: Capa de Aplicación.

7.7.1 Implementación de la aplicación del usuario

La aplicación de usuario se implementó en Android. Las tareas que realiza la aplicación son:

- Registrar los datos del usuario.
- Recibir notificaciones de alerta de cambios en la temperatura y la presión.
- Evaluar el estado de salud de los usuarios.

Registrar los datos del usuario

La aplicación permite registrar datos de usuario del sistema, para ello establece una interfaz de recolección de datos personales (nombre, apellidos, género y fecha de nacimiento) tal como se observa en la Figura 46. Estos datos se almacenan de manera

local y en la nube. El almacenamiento local de los datos se realiza mediante la clase SharedPreferences y esta permitirá personalizar la experiencia del usuario. El almacenamiento de los datos en la nube de Google sirve para que los modelos de Machine Learning creen predicciones individuales.



Figura 46: Interfaz de registro de usuario.

Recibir notificaciones de alerta de cambios en la temperatura y la presión.

La aplicación también permite recibir notificaciones de alerta cuando ocurre un evento de cambio en la temperatura y la presión atmosférica. Para determinar cuándo enviar una notificación al dispositivo del usuario, se recurre al modelo de predicción implementado en el servicio Cloud Machine Learning.

La notificación contiene un mensaje como el que aparece en la Figura 47 y dos opciones. La opción cancelar permite cerrar la notificación y la otra opción acepta iniciar la aplicación para evaluar el estado de salud del usuario.

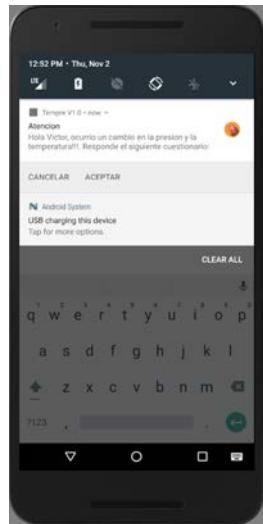


Figura 47: Notificación de alerta.

Evaluar el estado de salud de los usuarios.

La evaluación del estado de la salud de los usuarios se efectúa mediante dos cuestionarios de imágenes.

El primer cuestionario de la Figura 48, permite saber si el usuario siente dolor ante los cambios de temperatura y presión. Para obtener esta información se utilizó la escala facial de dolor de Wong Baker [65]. La escala facial de dolor permite identificar la intensidad de dolor de un usuario por medio de rostros. Cada rostro representa un estado diferente, desde la felicidad hasta la tristeza, según la intensidad de dolor:

- Rostro 0: No duele.
- Rostro 1: Duele muy poco.
- Rostro 2: El dolor es perceptible.
- Rostro 3: El dolo es molesto.
- Rostro 4: El dolo es intenso.
- Rostro 5: Máxima intensidad de dolor.

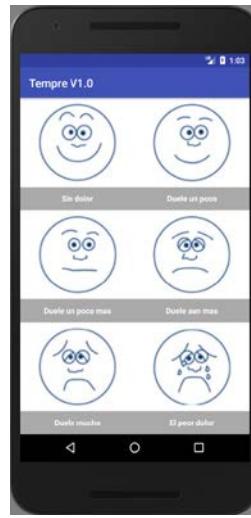


Figura 48: Cuestionario facial de dolor.

El segundo cuestionario de la Figura 49, permite identificar las regiones donde se presenta el dolor. En este cuestionario el usuario puede seleccionar las partes de su cuerpo donde se presenta el dolor. Para posteriormente finalizar enviando la información de la evaluación a la nube de Google.

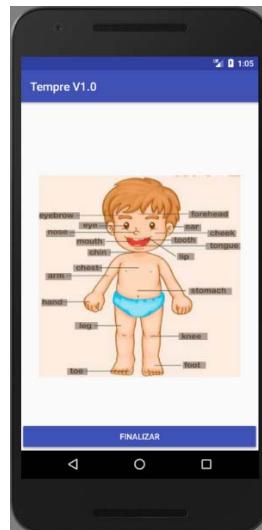
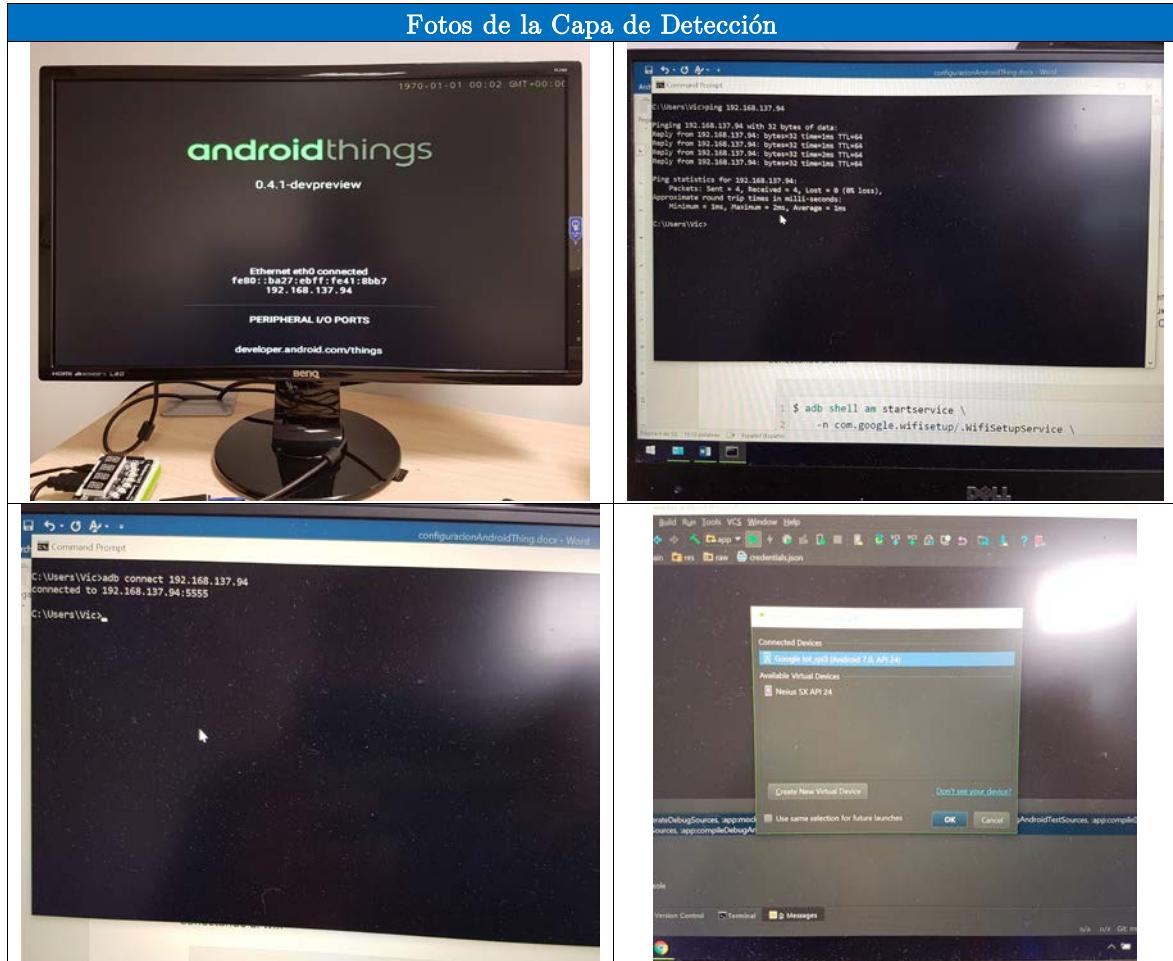


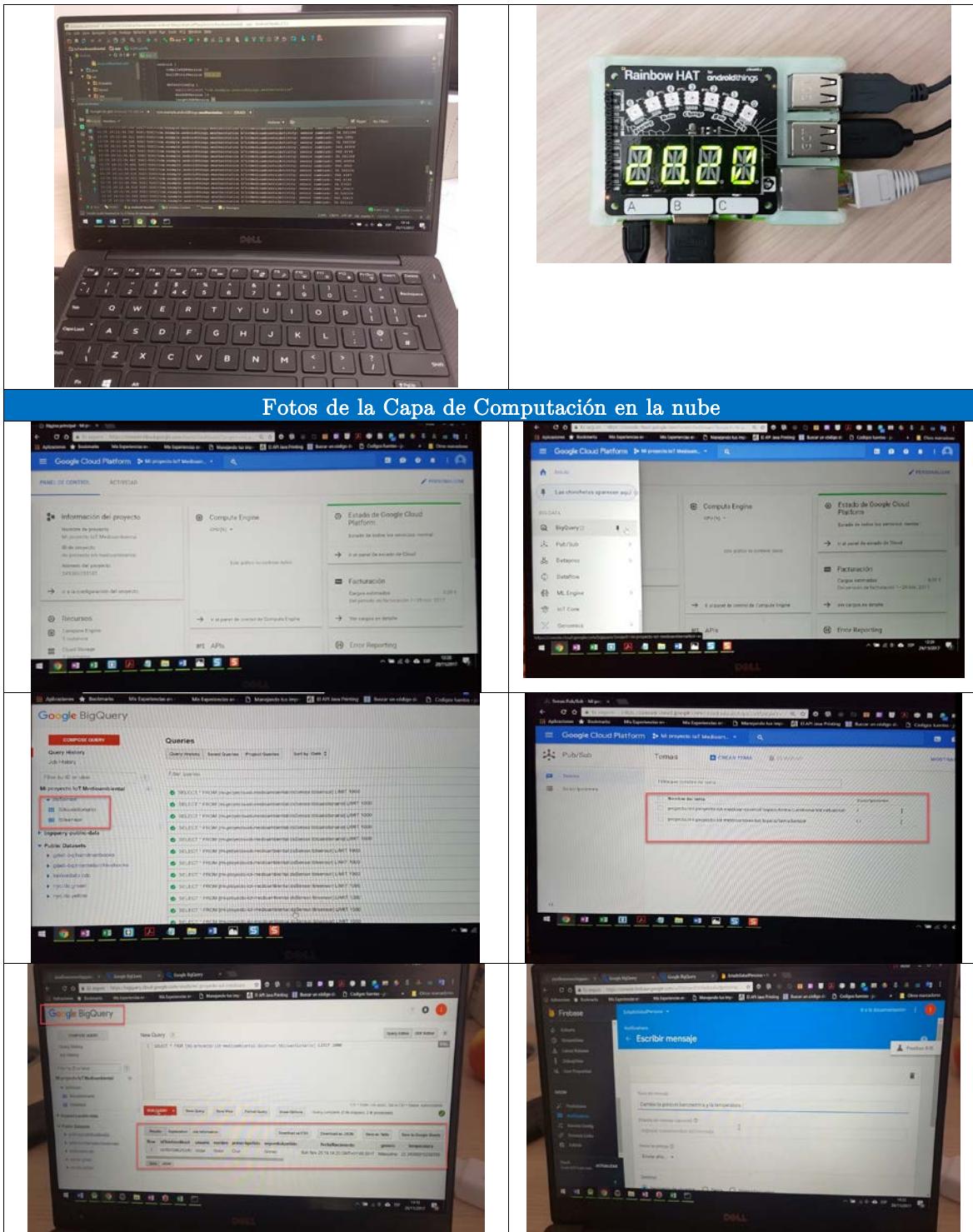
Figura 49: Cuestionario para la detección de regiones con dolor.

7.8 Fotografías del Sistema

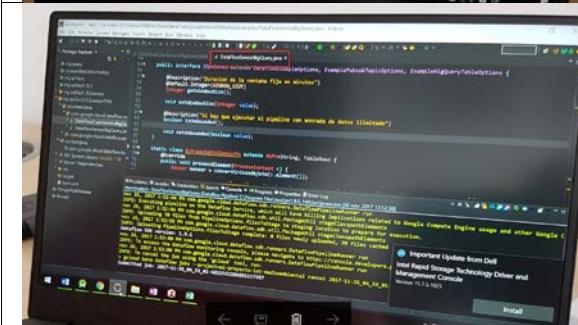
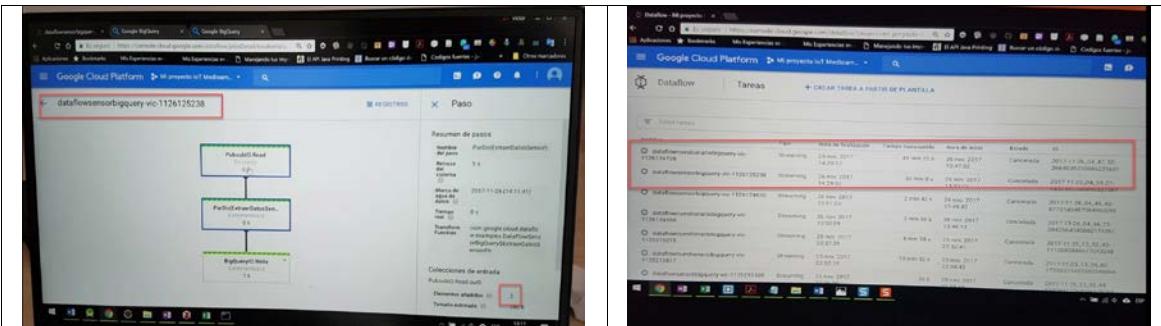
En la Tabla 33 se muestran las fotografías tomadas por cada Capa del Sistema Medioambiental IoT.

Tabla 33: Fotografías del Sistema medioambiental IoT.

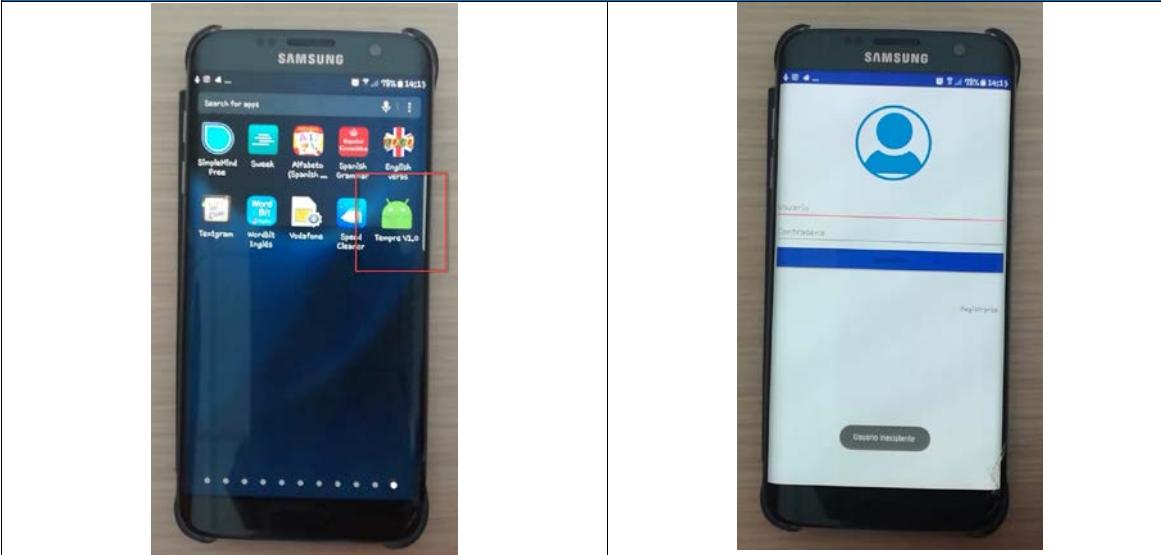


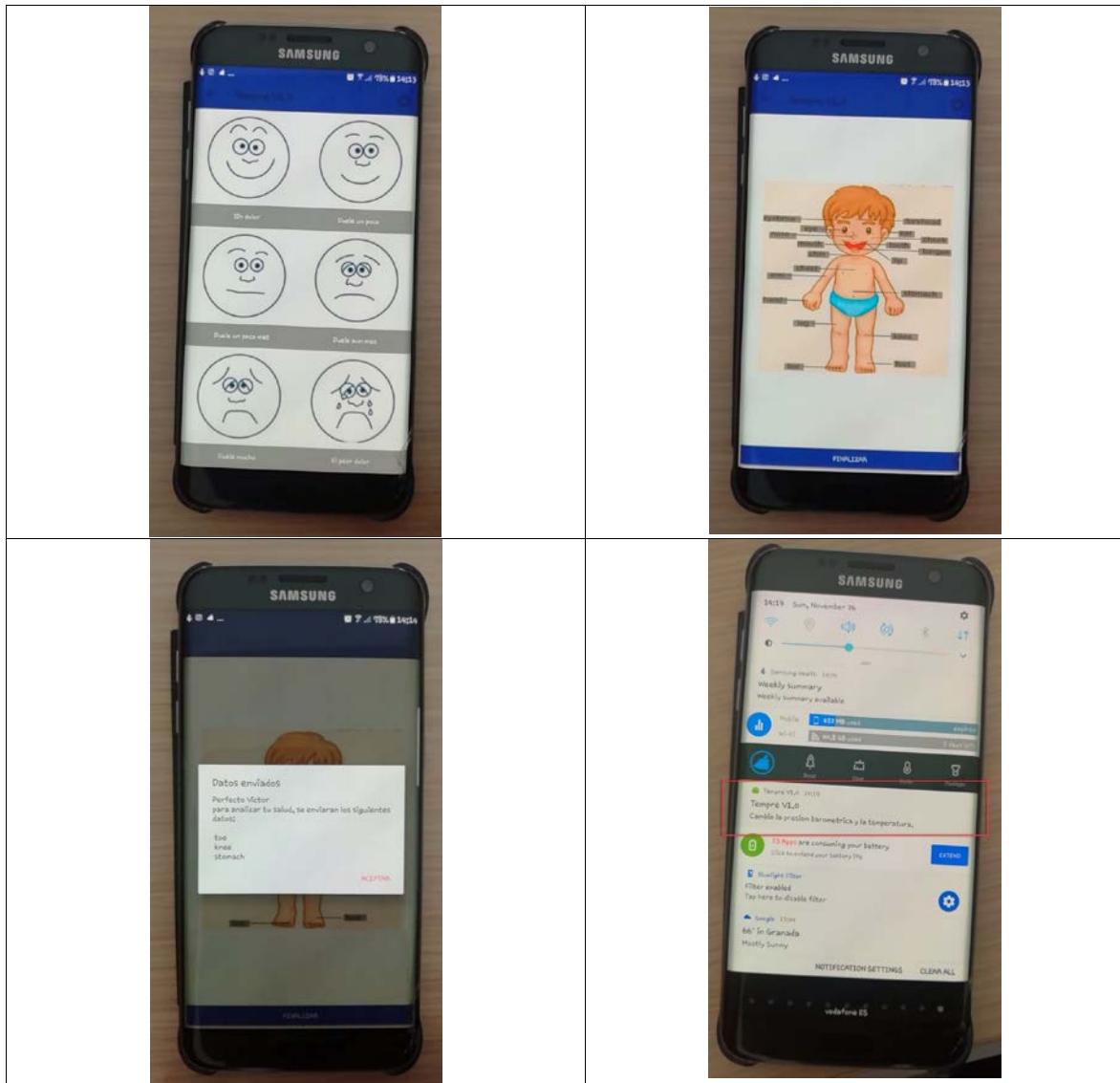


Fotos de la Capa de Computación en la nube



Fotos de la Capa de Aplicación





Capítulo 8

Conclusiones

8.1 Conclusiones

De acuerdo con el objetivo general de desarrollar un sistema medioambiental del IoT basado en tecnologías de Google, se cumplió dicho objetivo implementando una infraestructura del IoT que permite recolectar los datos medioambientales de temperatura y presión atmosférica para, posteriormente, procesar esta información en la plataforma de computación situada en la nube de Google.

Una vez cumplidos los objetivos específicos se puede concluir lo siguiente:

1. Recopilar información relacionada al Internet de las Cosas.

En lo que respecta a la recopilación de información referente al Internet de las Cosas, en el Capítulo 2 se compiló información relevante desde el año 2010 al 2016, de artículos publicados en las principales revistas científicas. De esta información se obtuvieron: antecedentes históricos relacionados a eventos tecnológicos, foros y conferencias desde el año 1961 al 2015; definiciones del Internet de las Cosas; se identificaron los elementos que componen un producto del IoT además de aportar propuestas arquitectónicas y tecnologías para el IoT.

2. Explorar las tecnologías del Internet de las Cosas que se utilizan actualmente.

Se investigaron las principales tecnologías del IoT y cada tecnología se clasificó por cada elemento del IoT tal como se observa en la Tabla 34. Esta información se detalla en el Capítulo 2, sección 2.6.

Tabla 34: Clasificación de tecnologías por elementos del IoT.

Elementos del IoT		Tecnologías
Identificación	Nombramiento	EPC, uCode
	Direccionamiento	IPv4, IPv6
Detección		Teléfonos inteligentes, Werables, Sensores, Actuadores, Sistemas Empotrados, Etiquetas RFID
Comunicación	Tecnologías de comunicación	Wifi, Bluetooth, Zigbee, SigFox, LoRa, NB-IoT
	Protocolos de comunicación	CoAP, MQTT, XMPP, AMQP, HTTP
Computación	Hardware	Raspberry PI, Arduino, Intel Edison, Intel Joule, versiones NXP, Phidgets, BeagleBone, Nanode
	Software	Contiki, Mbed OS, TynoOS, Micrium OS, RIOT, Brillo, Xively, Kaa, IBM Bluemix, Carriots, Nimbts

3. Estudiar la tecnología “Android Things” de Google, utilizado para el desarrollo de sistemas Empotrados.

Para cumplir este objetivo, se estudió la tecnología de Android Things y para ello en el Capítulo 4 se expusieron las características de este Sistema Operativo, se explicó su función en el IoT y se describió brevemente la evolución de esta tecnología. Además, se manifestó la arquitectura de Android Things y se identificaron los requisitos de Hardware y Software para su implementación. Así también se describieron importantes características tales como su compatibilidad con los servicios de google, seguridad etc. También en el Capítulo 4, se explicaron los controladores de usuario y su compatibilidad con periféricos de entrada y salida. Por último, se implementó una pequeña aplicación para encender y apagar un led, todo con el objetivo de mostrar brevemente la capacidad de conexión al Hardware desde una aplicación Android Things.

4. Estudiar el protocolo de comunicación “Weave” de Google, utilizado para la comunicación de los dispositivos del Internet de las Cosas.

Ya que en un principio se estableció como un objetivo específico y conociendo que el protocolo Weave forma parte del ecosistema de Google para el IoT, en el Capítulo 5 se describieron las características generales de este protocolo de comunicación, se

identificaron los requisitos de Hardware y Software para su implementación. Sin embargo, debido a las constantes actualizaciones en su documentación y en la consola del desarrollador Weave, no se logró implementar este protocolo en el proyecto. Como alternativa de comunicación se utilizaron las API de Google Cloud Platform las cuales permiten la comunicación entre los sensores y la plataforma de computación en la nube.

5. Estudiar la plataforma de computación en la nube “Google Cloud Platform” para la captura, procesamiento y almacenamiento de datos.

Como parte del estudio de la plataforma de computación en la nube “Google Cloud Platform”, en el Capítulo 6 se explicaron los servicios de esta plataforma de computación. Para ello se agruparon los diferentes servicios en componentes. Además, como parte del estudio de la plataforma de Google, se implementaron tres ejemplos básicos para mostrar su funcionamiento, para lo que se emplearon tres servicios utilizados para el IoT, estos son: Cloud Pub/Sub, Cloud Dataflow y el servicio de almacenamiento BigQuery.

6. Estudiar los efectos de la temperatura y la presión atmosférica en el estado de salud de las personas.

El objetivo principal de este proyecto fue construir un sistema del IoT medioambiental, para ello en la sección 7.1 del Capítulo 7, se compilaron artículos relacionados a los efectos que producen la temperatura y la presión atmosférica sobre la salud de las personas. Durante la recopilación de estos artículos se pudo constatar que existen varios estudios referentes a este tema. Por ejemplo, entre los artículos estudiados, destacamos artículos médicos relacionados con incidencias de hemorragia subaracnoidea -también conocida como hemorragia cerebral- causadas por la influencia de los cambios en la presión atmosférica. También se halló un estudio muy interesante que lo realizaron en la ciudad de Alicante (España), sobre los ingresos hospitalarios por enfermedades cardíacas y pulmonares, donde también determinaron que estos ingresos estaban relacionados con las variaciones de la densidad de oxígeno originados por los cambios en las masas de aire presente en la atmósfera.

7. Implementar un sistema empotrado para la obtención de la temperatura y presión medioambiental.

Respecto a este objetivo específico, como parte del desarrollo del presente proyecto, en la sección 7.5 del Capítulo 7, se logró desarrollar un Sistema Empotrado. Para ello se ensamblaron las placas de Hardware de Raspberry Pi y Rainbow HAT; este último es quien lleva integrados sensores de temperatura y presión atmosférica. Para su funcionamiento se creó una aplicación Java, la cual mediante el SDK de Android Things, permite obtener los datos de la temperatura y presión atmosférica. Terminada la creación de la aplicación, se procedió a desplegar el programa en el Sistema Empotrado que funciona con el sistema operativo Android Things, tal como se observa la Figura 50.



Figura 50: Sistema Empotrado en funcionamiento.

8. Implementar el protocolo Weave para la comunicación de los dispositivos del Internet de las Cosas.

Solo se describieron las características generales de este protocolo de comunicación y no se logró implementar el protocolo Weave por las frecuentes actualizaciones en la documentación y en la consola del desarrollador Weave. La implementación de la comunicación de los dispositivos se realizó mediante la API de comunicación provista por Google Cloud Platform.

9. Implementar una arquitectura del Internet de las Cosas en la infraestructura de Google Cloud Platform.

Como se explicó en la sección 7.3 del Capítulo 7, se logró establecer una arquitectura del IoT en la infraestructura de Google Cloud Platform, utilizando los servicios: Cloud Pub/Sub, Cloud Dataflow, Bigquery, Cloud Machine Learning, Compute Engine y Firebase Cloud Messaging tal como se observa en la Figura 51. Esta arquitectura diseñada para el presente proyecto medioambiental, podría ser aplicable a cualquier otro proyecto relacionado con el IoT, ya que en el mismo se consideran aspectos importantes para el procesamiento de grandes cantidades de datos -también denominado Big Data-. Sin embargo, si bien ya se encuentra implementado la arquitectura aún falta por completar el servicio Cloud Machine Learning ya que el mismo se encuentra orientado al análisis de la información para la obtención de patrones de datos.

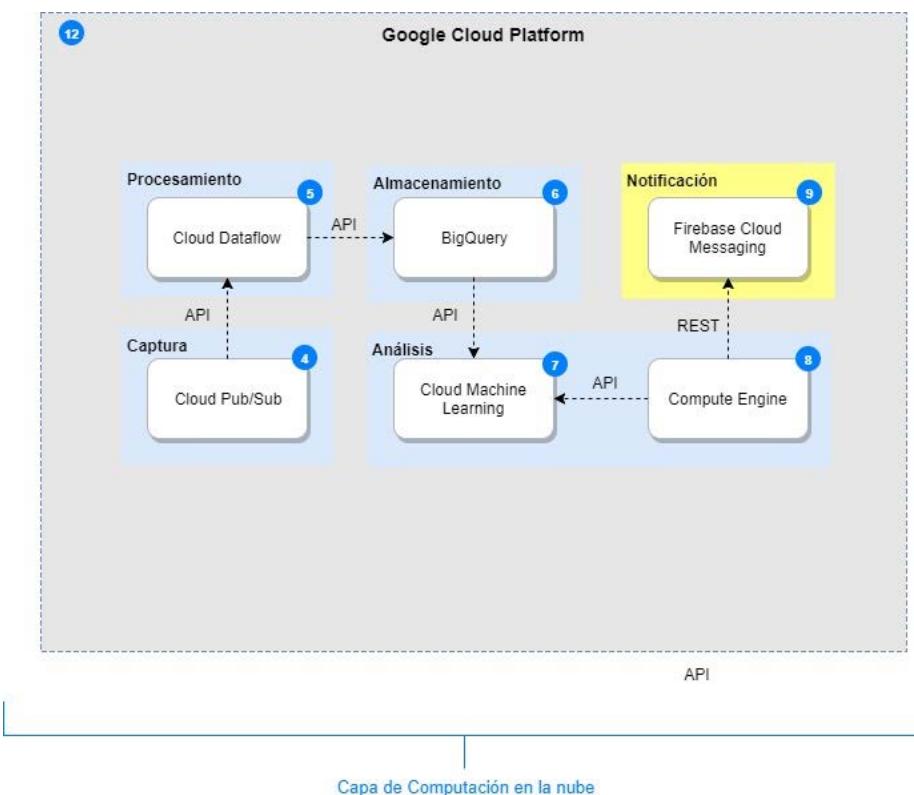


Figura 51: Capa de Computación en la nube.

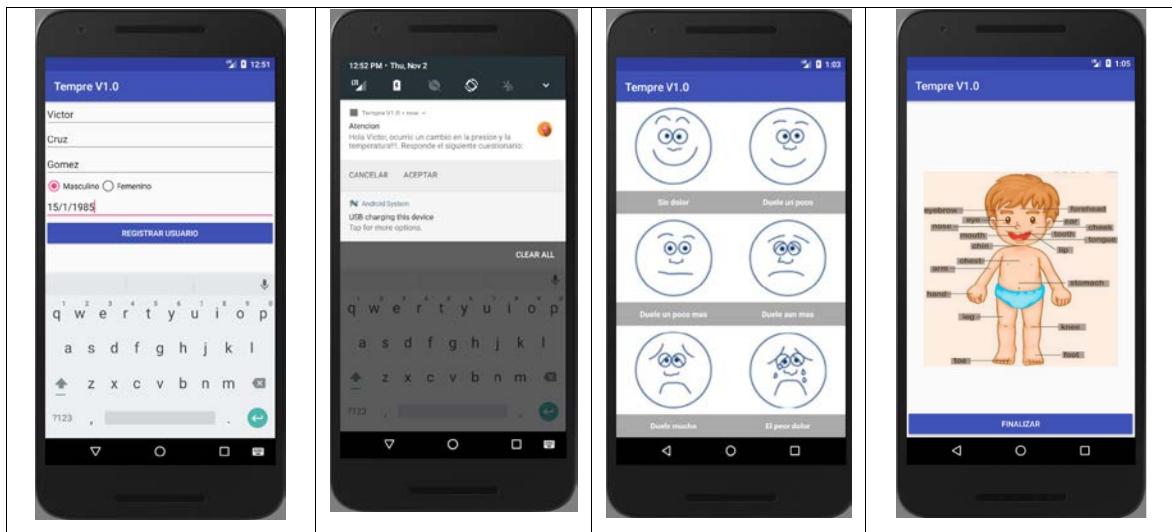
10. Integrar Android Things, Weave y Google Cloud Platform.

En un principio se tenía pensado trabajar con las dos aproximaciones, es decir IoT basado en Weave e IoT basado en Google Cloud Platform, pero por la falta de documentación en la plataforma Weave no se logró integrar Android Things con la misma. Sin embargo, sí que se lograron integrar tanto Android Things como Google Cloud Platform, por medio de la API del servicio Cloud Pub/Sub, tal como se explicó en la sección 7.3 del Capítulo 7. Por lo que se logró enviar mensajes desde dispositivos Android Things a la plataforma de computación en la nube.

11. Implementar una aplicación móvil para la evaluación del estado de salud de las personas.

Para lograr este objetivo, se desarrolló una aplicación móvil en Android con el objetivo de que el usuario final recibiera sugerencias y notificaciones desde la plataforma de Google. Igualmente, dentro de la aplicación se implementaron cuestionarios visuales con el objetivo de obtener el estado de salud de los usuarios; en concreto se crearon dos cuestionarios visuales. En el primer cuestionario se recurrió a los rostros de dolor para permitir identificar la intensidad de dolor que siente un usuario; para este fin se utilizó la escala facial de dolor de Wong Baker. El segundo cuestionario, nos permitió conocer las partes del cuerpo que pueden presentar dolor; para ello el usuario simplemente tendría que tocar una o varias partes donde el usuario sienta dolor, estos datos se recogen en la Tabla 35.

Tabla 35: Interfaces de la aplicación de usuario.



8.2 Problemas presentados

Los problemas presentados se refieren prácticamente a las dificultades técnicas presentadas durante la implementación del sistema medioambiental, que hemos agrupado de la siguiente forma:

Android Things

En un principio, la construcción del Sistema Empotrado se pensó realizar con sensores de la marca Grove y un módulo de extensión Grove Pi version 1.0. Los problemas que se presentaron al intentar reconocer los sensores por medio del módulo de extensión Grove Pi, se resumen principalmente en la falta de controladores por parte del fabricante Seed Studio, lo cual dificultó el reconocimiento de los sensores desde la aplicación desarrollada en Java. Por esta ausencia de disponibilidad de controladores, la integración se realizó mediante el módulo de Rainbow HAT, un módulo que incluye controladores para Android Things y varios sensores, entre ellos el de temperatura y presión atmosférica.

Otro problema al que nos enfrentamos, fue la conexión del Sistema Empotrado a la red. El sistema operativo Android Things aún no reconoce la comunicación inalámbrica Wifi y Bluetooth, por lo que no se trabajó con una comunicación inalámbrica sino con una conexión Ethernet. Sin embargo, también identificamos que al trabajar con una conexión Ethernet, a veces se pierde enlace entre el Sistema

Empotrado y el Router ya que existen dificultades en la asignación de IP por parte del servidor DHCP. Se detectó que estos problemas se deben a las versiones previas del sistema operativo ya que, el mismo aún no se encuentra estable por lo que cada 6 a 8 semanas, la compañía Google, lanza nuevas actualizaciones. Con respecto a las versiones previas de Android Things se recomienda trabajar con la versión previa 4.1 por ser un poco más estable.

Por último, otro problema presentado fue la falta de documentación técnica y códigos fuente para la programación de la aplicación de Android Things. No obstante, desde la fecha de lanzamiento de Android Things (diciembre del 2016) a día de hoy, ha mejorado la documentación oficial y ya se tienen aportes de códigos fuente de la comunidad de desarrolladores Android.

Weave

Los problemas presentados con este protocolo de comunicación giraron en torno a los constantes cambios en la estructura de su sitio web, en la documentación del protocolo y la falta de códigos fuentes para su implementación. Debido a los cambios en la estructura de su sitio web algunas páginas ya no se localizaron disponibles (error 404). La consola de desarrollador Weave móvil dejó de funcionar para convertirse en una aplicación web. En resumen, los constantes cambios en la documentación de Weave impidieron la implementación del protocolo en el presente proyecto.

Google Cloud Platform

De manera similar que, en los anteriores casos, en Google Cloud Platform se hallaron problemas en la documentación ya que estos no se encuentran muy detallados. Por lo que fue complicado realizar la integración se los servicios. También se encontraron problemas en las versiones de los SDK de los servicios Cloud Pub/Sub y Cloud Dataflow, puesto que en cada versión el código escrito no era compatible. Por ejemplo, en las notas de actualización de Google Cloud Message se menciona: “Dataflow sdk 2.x para Java no es compatible con Dataflow 1.x”. De manera similar la Migración de Google Cloud Message a Firebase Cloud Message también generó problemas, al igual que en los casos anteriores muchas funcionalidades de Google Cloud Message dejaron de trabajar. Por otro lado, los servicios en versión “beta private” generaron limitación de uso, por ejemplo: Google Cloud IoT Core no se logró utilizar porque el acceso está limitado a empresas.

Android

En la aplicación del cliente, desarrollado en Android, se presentaron problemas en la integración de la API del servicio Cloud Pub/Sub por la falta de una documentación técnica detallada.

En general, los problemas presentados al trabajar con el ecosistema de tecnologías de Google se debieron a: la falta de documentación detallada, a las versiones beta y a las constantes actualizaciones en sus tecnologías por lo que, en consecuencia, todo ello complicó la integración de las tecnologías utilizadas. A pesar ello se logró implementar el sistema medioambiental.

8.3 Análisis de los objetivos

En el Gráfico 1 se observa el grado de consecución de cada uno de los objetivos específicos que se habían planteado en el trabajo.

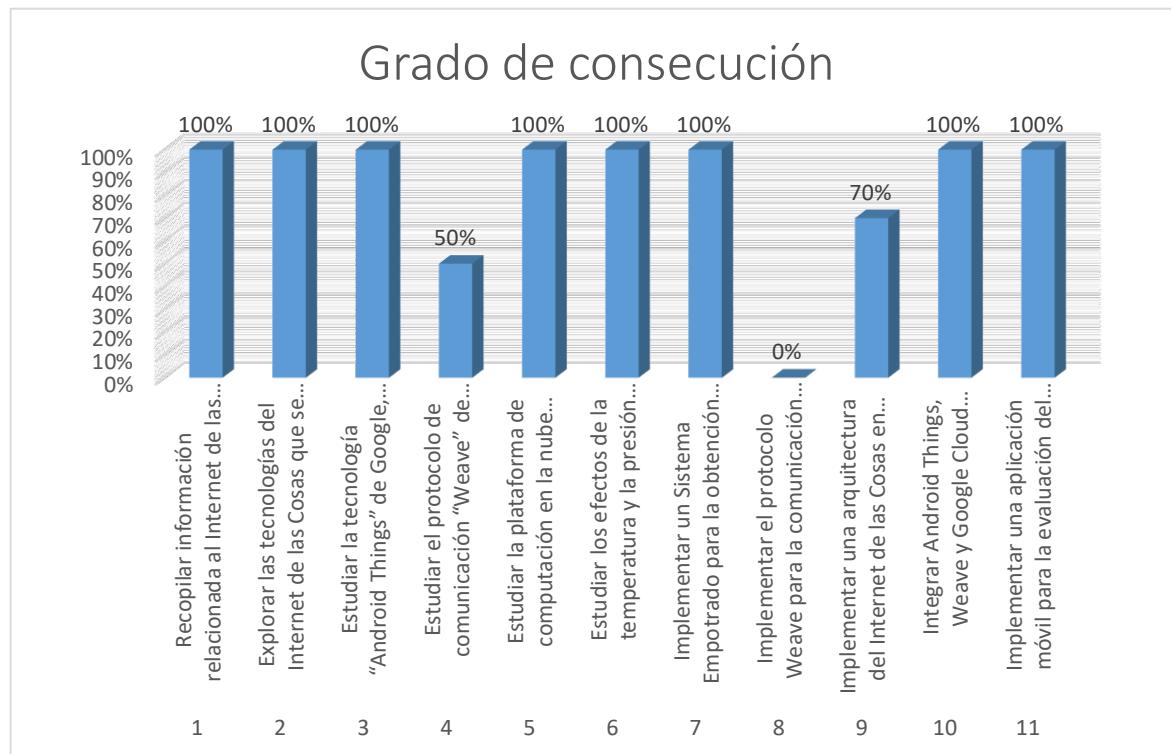


Gráfico 1: Grado de consecución de cada objetivo específico.

8.4 Trabajos futuros

Para establecer líneas futuras de desarrollo a modo de continuación de este trabajo fin de máster y para el adecuado enfoque sistémico del sistema medioambiental: quedan abiertas algunas actividades por completar que han ido surgiendo a lo largo de este proyecto y que habrán de ser desarrolladas en un futuro. Entre estas posibles actividades destacamos las siguientes:

Capa de detección

- Implementar las actualizaciones OTA, para actualizar el sistema operativo Android Things. Las actualizaciones OTA se realizan desde la consola de desarrolladores Android Things.
- Habilitar la conexión inalámbrica por Wifi.
- Adicionar un módulo de GPS para conocer la ubicación del dispositivo medioambiental.
- Implementar el protocolo Weave o MQTT por el tamaño de sus paquetes de mensajes.
- Utilizar la plataforma de Hardware Raspberry PI Zero para reducir el tamaño del Sistema Empotrado.

Capa de computación

- Implementar el servicio Cloud IoT Core, para la comunicación bidireccional entre el Sistema Empotrado y el Servicio Cloud IoT Core.
- Implementar el servicio Cloud Machine Learning para generar modelos de aprendizaje automáticos.
- Utilizar Datalab para el análisis y la visualización de datos, pues se conecta fácilmente a otros servicios para centrarse en las tareas relacionadas con la ciencia de datos. También permite usar la biblioteca TensorFlow para la creación de modelos de aprendizaje automático.
- Integrar la API AccuWeather API para cruzar la información medioambiental recolectado por los sensores con el pronóstico del tiempo proporcionado por la API.

Capa de Aplicación

- Implementar la aplicación de usuario final para sistemas operativos móviles como: IOS, Windows Phone y otros.
- Diseñar una aplicación adaptable según el género del usuario. El nuevo diseño debería de adaptarse según el género y la edad del usuario para facilitar la usabilidad del mismo.

Referencias Bibliográficas

- [1] J. Donica, *Pax Technica: How the Internet of things may set us free or lock us up. Philip N. Howard*. Oxford University Press, 2017.
- [2] D. Evans, «Internet de las Cosas como la proxima evolucion de Internet lo cambia todo.», *Cómo Próxima Evol. Internet Lo Cambia Todo Cisco Internet Bussiness Solut. Group-IBSG*, vol. 11, n.º 1, pp. 4–11, 2011.
- [3] «La nueva revolución industrial: el internet de las cosas | economía | EL MUNDO», *ELMUNDO*, 30-jun-2014. [En línea]. Disponible en: <http://www.elmundo.es/economia/2014/06/30/53b14869e2704e97368b4577.html>. [Accedido: 10-oct-2017].
- [4] «Microsoft Azure: plataforma y servicios de informática en la nube». [En línea]. Disponible en: <https://azure.microsoft.com/es-es/>. [Accedido: 10-oct-2017].
- [5] «Windows 10 IoT Core Official Website | Developer Resource | Windows IoT». [En línea]. Disponible en: <https://developer.microsoft.com/es-es/windows/iot>. [Accedido: 10-oct-2017].
- [6] «Descargue Packet Tracer | Cisco NetAcad». [En línea]. Disponible en: <https://www.netacad.com/es/courses/packet-tracer-download/>. [Accedido: 10-oct-2017].
- [7] «Smart home - Smartthings | Samsung US», *Samsung Electronics America*. [En línea]. Disponible en: <http://www.samsung.com/us/smart-home/smartthings/>. [Accedido: 10-oct-2017].
- [8] «Comprar Philips Hue Iluminación inalámbrica personal 8718291241751 Iluminación inalámbrica personal», *Philips*. [En línea]. Disponible en: <https://www.philips.es/c-p/8718291241751/hue-iluminacion-inalambrica-personal>. [Accedido: 10-oct-2017].
- [9] «Announcing updates to Google's Internet of Things platform: Android Things and Weave», *Android Developers Blog*.
- [10] L. Atzori, A. Iera, y G. Morabito, «The Internet of Things: A survey», *Comput. Netw.*, vol. 54, n.º 15, pp. 2787–2805, oct. 2010.
- [11] M. J. Castro Rocha, «Incidencia de las hemorragias subaracnoidea aneurismática y la influencia de los cambios de la presión atmosférica en los pacientes ingresados en el servicio de neurocirugía del Hospital Antonio Lenin Fonseca en el periodo de Enero del 2014 a Diciembre del 2015», other, Universidad Nacional Autónoma de Nicaragua, UNANManagua, 2016.
- [12] F. J. M. Gaona, F. L. T. Gonçalves, A. S. Nedel, y R. C. Alves, «Condiciones ambientales externas y su relación con la morbilidad infantil: estudio de caso, São Paulo, Brasil», *Épsilon*, n.º 21, pp. 107–118, 2014.
- [13] J. Olcina Cantos, D. Martín Estévez, y others, «Variaciones en la densidad del oxígeno en el aire y su influencia sobre la salud humana», 2012.
- [14] A. Paolasso, «Diestres Metereologico», 1999.
- [15] «Cuando el cuerpo pronostica el tiempo», *La Vanguardia*. [En línea]. Disponible en:

- <http://www.lavanguardia.com/estilos-de-vida/20120427/54285747599/cuando-el-cuerpo-pronostica-el-tiempo.html>. [Accedido: 16-nov-2017].
- [16] M. C. Herrara, «Percepciones sobre los fenómenos de variabilidad climática y cambio climático entre campesinos del centro de Santander, Colombia - ProQuest», dic. 2012.
 - [17] P. Rebollo Dujisin, «MAL AGUDO DE MONTAÑA Y EXPERIENCIA TURÍSTICA. La deuda del turismo en Chile», *Gest. Tur.*, n.º 16, 2011.
 - [18] K. Ashton, «That ‘internet of things’ thing», *RFID J.*, vol. 22, n.º 7, 2011.
 - [19] J. Gubbi, R. Buyya, S. Marusic, y M. Palaniswami, «Internet of Things (IoT): A vision, architectural elements, and future directions», *Future Gener. Comput. Syst.*, vol. 29, n.º 7, pp. 1645–1660, 2013.
 - [20] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, y M. Ayyash, «Internet of things: A survey on enabling technologies, protocols, and applications», *IEEE Commun. Surv. Tutor.*, vol. 17, n.º 4, pp. 2347–2376, 2015.
 - [21] M. Gigli y S. Koo, «Internet of Things: Services and Applications Categorization», *Adv. Internet Things*, vol. 01, n.º 02, pp. 27-31, 2011.
 - [22] S. Sankar y P. Srinivasan, «INTERNET OF THINGS (IOT): A SURVEY ON EMPOWERING TECHNOLOGIES, RESEARCH OPPORTUNITIES AND APPLICATIONS».
 - [23] L. Tan y N. Wang, «Future internet: The internet of things», en *Advanced Computer Theory and Engineering (ICACTE), 2010 3rd International Conference on*, 2010, vol. 5, pp. V5–376.
 - [24] S. Fang *et al.*, «An integrated system for regional environmental monitoring and management based on internet of things», *IEEE Trans. Ind. Inform.*, vol. 10, n.º 2, pp. 1596–1605, 2014.
 - [25] T. Malche y P. Maheshwary, «Harnessing the Internet of Things (IoT): A Review», *Int. J.*, vol. 5, n.º 8, 2015.
 - [26] D. Engels, «The use of the electronic product code», *Auto-ID Cent. Mass. Inst. Technol. Tech. Rep.*, 2003.
 - [27] U. I. Center, *Ubiquitous Code: ucode*. ed, 2009.
 - [28] A. N. A. Ali, «Comparison study between IPV4 & IPV6», *Int. J. Comput. Sci. Issues*, vol. 9, n.º 3, pp. 314–317, 2012.
 - [29] «Definición de smartphone — Definicion.de», *Definición.de*. [En línea]. Disponible en: <https://definicion.de/smartphone/>. [Accedido: 18-oct-2017].
 - [30] «wearable Meaning in the Cambridge English Dictionary». [En línea]. Disponible en: <http://dictionary.cambridge.org/dictionary/english/wearable>. [Accedido: 18-oct-2017].
 - [31] «sensor - Definición - WordReference.com». [En línea]. Disponible en: <http://www.wordreference.com/definicion/sensor>. [Accedido: 18-oct-2017].
 - [32] V. Pérez, «INGENIERÍA PARA EL REEMPLAZO DE ACTUADORES YUGO ESCOCÉS POR ACTUADORES GAS-HIDRÁULICOS, EN ESTACIONES DE VÁLVULAS AUTOMÁTICAS», Universidad de Oriente, 2009.
 - [33] D. I. Tapia, J. R. Cueli, Ó. García, J. M. Corchado, J. Bajo, y A. Saavedra,

- «Identificacion por radiofrecuencia: fundamentos y aplicaciones», *Jorn. Científicas Sobre RFID*, 2007.
- [34] V. Abinayaa y A. Jayan, «Case study on comparison of wireless technologies in industrial applications», *Int. J. Sci. Res. Publ.*, vol. 4, n.º 2, pp. 1–4, 2014.
 - [35] «Sigfox Technology Overview | Sigfox». [En línea]. Disponible en: <https://www.sigfox.com/en/sigfox-iot-technology-overview>. [Accedido: 17-oct-2017].
 - [36] «lora-alliance | Technology», *lora-alliance*. [En línea]. Disponible en: <https://www.lora-alliance.org/technology>. [Accedido: 17-oct-2017].
 - [37] «que es gsma - Buscar con Google». [En línea]. Disponible en: <https://www.google.es/search?q=que+es+gsma&oq=que+es+gsma&aqs=chrome..69i57j0l4.2153j0j1&sourceid=chrome&ie=UTF-8>. [Accedido: 17-oct-2017].
 - [38] V. Karagiannis, P. Chatzimisios, F. Vazquez-Gallego, y J. Alonso-Zarate, «A survey on application layer protocols for the internet of things», *Trans. IoT Cloud Comput.*, vol. 3, n.º 1, pp. 11–17, 2015.
 - [39] D. Guinard, V. Trifa, F. Mattern, y E. Wilde, «From the internet of things to the web of things: Resource-oriented architecture and best practices», *Archit. Internet Things*, pp. 97–129, 2011.
 - [40] «Raspberry Pi - Teach, Learn, and Make with Raspberry Pi», *Raspberry Pi*. [En línea]. Disponible en: <https://www.raspberrypi.org/>. [Accedido: 16-oct-2017].
 - [41] «Arduino - Home». [En línea]. Disponible en: <https://www.arduino.cc/>. [Accedido: 16-oct-2017].
 - [42] «Asistencia para la Placa Intel® Edison para Arduino*», *Intel*. [En línea]. Disponible en: <https://www.intel.com/content/www/xl/es/support/products/84574/boards-and-kits/intel-edison-boards/intel-edison-board-for-arduino.html>. [Accedido: 16-oct-2017].
 - [43] «Especificaciones de producto de Intel® Joule™ 570x Developer Kit», *Intel® ARK (Product Specs)*. [En línea]. Disponible en: <https://ark.intel.com/es-es/products/96414/Intel-Joule-570x-Developer-Kit>. [Accedido: 16-oct-2017].
 - [44] «Internet of Things (IoT)|NXP». [En línea]. Disponible en: <https://www.nxp.com/applications/solutions/internet-of-things:Internet-of-Things-IoT>. [Accedido: 16-oct-2017].
 - [45] «Phidgets Inc. - Products for USB Sensing and Control». [En línea]. Disponible en: <https://www.phidgets.com/>. [Accedido: 16-oct-2017].
 - [46] «BeagleBoard.org - bone». [En línea]. Disponible en: <http://beagleboard.org/bone>. [Accedido: 16-oct-2017].
 - [47] «Nanode», *Nanode*. [En línea]. Disponible en: <http://www.nanode.eu/>. [Accedido: 16-oct-2017].
 - [48] «Liberium - Connecting Sensors to the Cloud». [En línea]. Disponible en: <http://www.libelium.com/>. [Accedido: 16-oct-2017].
 - [49] «Android Things | Android Things». [En línea]. Disponible en: <https://developer.android.com/things/hardware/index.html>. [Accedido: 15-nov-2017].
 - [50] «Release Notes Nest Weave», *Nest Developers*. [En línea]. Disponible en:

- <https://developers.nest.com/documentation/cloud/release-notes>. [Accedido: 20-nov-2017].
- [51] «Cloud Computing, servicios de alojamiento y APIs de Google Cloud», *Google Cloud Platform*. [En línea]. Disponible en: <https://cloud.google.com/?hl=es>. [Accedido: 15-nov-2017].
 - [52] «Google Cloud Platform». [En línea]. Disponible en: <https://console.cloud.google.com/start?hl=es-419>. [Accedido: 05-nov-2017].
 - [53] «Google closes \$3.2 billion purchase of Nest», *CNET*. [En línea]. Disponible en: <https://www.cnet.com/news/google-closes-3-2-billion-purchase-of-nest/>. [Accedido: 28-nov-2017].
 - [54] posts > F. W. 1227, «Android Things Developer Preview 2 Released – Tensorflow, Peripheral I/O Among Lib's Added», *xda-developers*, 09-feb-2017. .
 - [55] «Google I/O 2015: Project Brillo, o el internet de las cosas | Mobility». [En línea]. Disponible en: <http://sevilla.abc.es/mobility/noticia/android/noticias-android/google-io-2015-project-brillo-o-el-internet-de-las-cosas/>. [Accedido: 15-nov-2017].
 - [56] «Overview | Android Things». [En línea]. Disponible en: <https://developer.android.com/things/sdk/index.html>. [Accedido: 20-nov-2017].
 - [57] «Release Notes | Android Things». [En línea]. Disponible en: <https://developer.android.com/things/preview/releases.html>. [Accedido: 20-oct-2017].
 - [58] «Google». [En línea]. Disponible en: <https://www.google.es/>. [Accedido: 23-nov-2017].
 - [59] «Google opens up Brillo and Weave to more developers, EFF wins car software petition, and the iOS 9.2 beta—SD Times news digest: Oct. 28, 2015», *SD Times*, 28-oct-2015. .
 - [60] Nest, «Notas de la version Weave.», 2017.
 - [61] *DataflowJavaSDK: Google Cloud Dataflow provides a simple, powerful model for building both batch and streaming parallel data processing pipelines*. Google Cloud Platform, 2017.
 - [62] «Google Cloud Storage Authentication | Cloud Storage Documentation», *Google Cloud Platform*. [En línea]. Disponible en: <https://cloud.google.com/storage/docs/authentication>. [Accedido: 04-nov-2017].
 - [63] «Beam Programming Guide». [En línea]. Disponible en: <https://beam.apache.org/documentation/programming-guide/>. [Accedido: 05-nov-2017].
 - [64] «Firebase Cloud Messaging», *Firebase*. [En línea]. Disponible en: <https://firebase.google.com/docs/cloud-messaging/?hl=es-419>. [Accedido: 03-nov-2017].
 - [65] B. Wong, «Wong Baker Faces», *Wong-Baker FACES Foundation*. [En línea]. Disponible en: <http://wongbakerfaces.org/>. [Accedido: 08-oct-2017].

Anexos

Anexo A - Guías para dispositivos Weave

Tabla 36: Guías para dispositivos Weave - bombillas.

Bombillas	
Funciones Genéricas	Estados
Device: describe la información del dispositivo como ser: fabricante, modelo.	<ul style="list-style-type: none">Nombre del dispositivo.Ubicación.Descripción del dispositivo.Número de serie.ID generado por el dispositivo utilizado para la identificación en redes locales y el acceso sin conexión.Versión del firmware del dispositivo.Errores (valor fuera de rango, valor invalido).
OnOff: describe el estado de la luz.	<ul style="list-style-type: none">Estado de encendido o apagado.Errores (error inesperado, valor invalido).
Brightness: describe el nivel de irradiación de la luz.	<ul style="list-style-type: none">Brillo de la luz entre 0 y 1.Errores (error inesperado al cambiar el estado, valor fuera de rango).
ColorXy: describe el color de luz implementada como un punto en el espacio.	<ul style="list-style-type: none">Potencia del color de luz para los colores rojo, verde y azul.Errores (error inesperado al cambiar el estado, valor fuera de rango).
ColorTemp: describe la temperatura de color de una fuente de luz.	<ul style="list-style-type: none">Temperatura del color.Temperatura del color mínimo y máximo.Errores (error inesperado al cambiar el estado, valor fuera de rango).

Tabla 37: Guías para dispositivos Weave - tomacorrientes.

TomaCorrientes	
Funciones Genéricas	Estados
Device: describe la información del dispositivo como ser: fabricante, modelo.	<ul style="list-style-type: none"> Nombre del dispositivo. Ubicación. Descripción del dispositivo. Número de serie. ID generado por el dispositivo utilizado para la identificación en redes locales y el acceso sin conexión. Versión del firmware del dispositivo. Errores (valor fuera de rango, valor invalido).
OnOff: describe el estado de alimentación del tomaCorriente.	<ul style="list-style-type: none"> Estados de encendió o apagado. Errores (valor invalido).
Brightness: describe la energía variable entregado al dispositivo conectado en el tomaCorriente, por ejemplo, regulación de la luz de una lámpara conectado al tomaCorriente.	<ul style="list-style-type: none"> Luminosidad de la fuente de luz. Errores (valore fuera de rango).

Tabla 38: Guías para dispositivos Weave - televisores.

Televisores	
Funciones Genéricas	Estados
Device: describe la información del dispositivo como ser: fabricante, modelo.	<ul style="list-style-type: none"> Nombre del dispositivo. Ubicación. Descripción del dispositivo. Número de serie. ID generado por el dispositivo utilizado para la identificación en redes locales y el acceso sin conexión. Versión del firmware del dispositivo. Errores (valor fuera de rango, valor invalido).
OnOff: describe el estado de alimentación del televisor.	<ul style="list-style-type: none"> Estados de encendido o apagado. Errores (valor invalido).
VolumenSetting: describe el estado del volumen del televisor.	<ul style="list-style-type: none"> Ajuste del volumen del televisor. Verificar si el televisor está o no

	<ul style="list-style-type: none"> enmudecido. • Errores (valor fuera de rango, estado inválido).
VideoInputs: describe los tipos de entrada de video disponibles en el televisor.	<ul style="list-style-type: none"> • InputType: Lista de posibles tipos de entrada. • InputState: Colección de todos los estados de entrada de video definidos o declarados por el fabricante (HDMI, USB, SVIDEO, etc.). • InputConfig: Comando para cambiar todos los caracteres de escritura en el estado de entrada. • CurrentInput: Identifica que entradas de video es la entrada activa.

Tabla 39: Guías para dispositivos Weave - interruptores.

Interruptores	
Funciones Genéricas	Estados
Device: describe la información del dispositivo como ser: fabricante, modelo.	<ul style="list-style-type: none"> • Nombre del dispositivo • Ubicación • Descripción del dispositivo • Número de serie. • ID generado por el dispositivo utilizado para la identificación en redes locales y el acceso sin conexión. • Versión del firmware del dispositivo. • Errores (valor fuera de rango, valor invalido)
OnOff: describe el estado de alimentación de un interruptor.	<ul style="list-style-type: none"> • Estados de encendido o apagado • Errores (valor invalido)

Anexo B - Google Cloud Platform

Tabla 40: Servicios de Google Cloud Platform.

Google Cloud Platform	Productos y Servicios	Descripción
Identidad y Seguridad	Cloud IAM	Cloud IAM (Gestión de acceso de identidad) permite gestionar los usuarios, el acceso y la visibilidad que tendrán sobre los recursos. Gestiona también de forma centralizada todos los recursos (servicios, cuentas de usuarios, permisos) de la empresa online y controla el acceso a proyectos de Google Cloud Platform mediante políticas de seguridad e integra auditorias para facilitar su cumplimiento.
	Cloud Identity Aware Proxy (beta)	El Cloud Identity Aware Proxy (Cloud IAP) controla el acceso a aplicaciones en la nube mediante la verificación de la identidad del usuario. El Cloud IAP es un servicio que implementa un modelo de seguridad de Google llamado BeyondCorp que permite a todos los empleados trabajar desde redes no confiables sin el uso de una VPN.
	API de Cloud Data Loss Prevention	La API de Cloud Data Loss Prevention (Prevención de perdida de datos) permite gestionar los datos sensibles de una organización. Es una estrategia para asegurarse que los usuarios finales no envíen información sensible o crítica fuera de la red corporativa. La API puede ser útil para identificar contenido bien definido como los números de seguro social o tarjetas de crédito.
	Security Key Enforcement	Security Key Enforcement, es un dispositivo físico que sirve para iniciar sesión. Se conecta en el puerto USB de la computadora. Las claves de seguridad se generan al pulsar el botón del dispositivo físico. A diferencia de otros métodos de verificación que utilizan códigos a través de mensajes de texto, este no requiere vincular un número de teléfono a una cuenta.
	Cloud Key Management Service	Cloud Key Management Service es un servicio de administración de claves alojado en la nube. Utiliza el algoritmo de cifrado AES -256 (Advanced Encryption Standard). Cifra y descifra a través de una API REST.
	Cloud Resource Manager	Cloud Resource Manager proporciona contenedores de recursos (como carpetas o proyectos) que permiten agrupar y organizar jerárquicamente otros recursos de Cloud Platform.
	Cloud	Cloud Security Scanner es una herramienta de análisis de

	Security Scanner	seguridad web que detecta las vulnerabilidades más comunes [XSS (Cross site scripting), introducción de flash y bibliotecas obsoletas que no son seguras] de las aplicaciones de Google App Engine.
Herramientas de Administración	Stackdriver	Stackdriver ofrece potentes funciones de depuración, alertas, registros, trazas e informes de errores de las aplicaciones de Cloud Platform y AWS. Dispone de integración nativa en las herramientas de datos de Google Cloud, como BigQuery, Cloud Pub/Sub, Cloud Storage y Cloud Datalab
	Monitoring	Monitoring permite ver con mayor calidad el rendimiento, el tiempo de funcionamiento y el estado general de las aplicaciones en la nube.
	Logging	Logging permite almacenar los datos y eventos de los registros de Google Cloud Platform y Amazon Web Services, buscarlos, analizarlos, supervisarlos y recibir alertas.
	Error Reporting	Error Reporting cuenta, analiza y agrupa los fallos de los servicios que se estén ejecutando en la nube.
	Trace	Trace es un sistema de rastreo que recopila datos de latencia de tus aplicaciones y los muestra en la consola de Google Cloud Platform. Permite hacer un seguimiento de cómo se propagan las solicitudes hacia una determinada apelación y recibir información detallada en tiempo real.
	Debugger	Es una característica de Google Cloud Platform para inspeccionar el estado de una aplicación en cualquier ubicación del código sin ralentizar las aplicaciones.
	Cloud Deployment Manager	Crea y administra recursos en la nube con plantillas sencillas de despliegue. Una característica importante de Cloud Deployment Manager es tratar la configuración como si fuera código lo cual permite efectuar despliegues repetibles.
	Cloud Endpoints	Cloud Endpoints es un servicio de gestión de API que se utiliza para alojar, proteger y controlar cualquier API.
	Cloud Console	Cloud Console es una interfaz de administración web mediante el cual se gestiona toda la información de los elementos que sostienen una aplicación en la nube (aplicaciones web, máquinas virtuales, almacenes y base de datos, análisis de datos, redes y servicios de desarrollo).
	Cloud Shell	Cloud Shell es una interfaz web que permite el acceso a recursos

		en la nube mediante línea de comandos.
Cloud Mobile App	Cloud Mobile App es una aplicación móvil mediante el cual se administra todos los servicios de Google Cloud Platform.	
API Billing de Cloud	API Billing de Cloud proporciona métodos para administrar la facturación de Google Cloud Platform mediante programación.	
API de Cloud	API de Cloud gestiona las interfaces de programación para todos los servicios de Google Cloud Platform. Permite acceder a los productos de Google Cloud Platform desde código. Además, permite automatizar los flujos de trabajo utilizando los lenguajes soportados por la plataforma (Java, Python, Go, C, Php). También hace posible el acceso a las diferentes API a través de llamadas REST.	
SDK de Cloud	SDK de Cloud es una interfaz de línea de comandos para productos y servicios. SDK de Cloud es un conjunto de comandos (gcloud, gsutil, bq, cmdlets, kubectl) que ofrecen acceso desde línea de comandos a Compute Engine, Cloud Storage, BigQuery y otros productos y servicios.	
Container Registry	Container Registry es un repositorio Docker privado. Permite almacenar imágenes Docker en Google Cloud Platform para recuperarlas y usarlas de forma rápida.	
Container Builder	Container Builder es una herramienta para construir imágenes de contenedores en la infraestructura de Google.	
Cloud Source Repository	Cloud Source Repository permite crear repositorios GIT privados alojados en Google Cloud Platform. Los repositorios de Google Cloud Source proporcionan control sobre las versiones de GIT para facilitar el desarrollo colaborativo.	
Cloud Tools para Android Studio	Google Cloud Platform suministra complementos y frameworks para Android Studio, el cual permite añadir fácilmente <i>back-ends</i> basados en la nube.	
Cloud Tools para IntelliJ	Complemento para depurar aplicaciones en la nube desde el IDE IntelliJ IDEA.	
Cloud Tools para PowerShell	PowerShell es una consola de sistema del Sistema Operativo Windows bastante avanzado. Cloud Tools para PowerShell ofrece la posibilidad de crear secuencias de comandos, automatizar y administrar las cargas de trabajo de Windows que se ejecutan en Cloud Platform.	
Cloud Tools	Es un complemento para el IDE Visual Studio, permite	

	para Visual Studio	desplegar aplicaciones en Google Cloud Platform.
	Cloud Tools para Eclipse	Es un complemento para el IDE Eclipse, permite desplegar aplicaciones Java en Google Cloud Platform.
	Complemento de Gradle para App Engine	Complemento Gradle para proyecto de App Engine.
	Complemento Maven para App Engine	Complemento Maven para proyecto de App Engine.
	Cloud Test Lab	Ofrece una infraestructura basada en la nube para probar aplicaciones de Android. Los resultados de las pruebas de las aplicaciones incluyen capturas de pantallas, videos y mapas de actividades.

Anexo C - Herramientas y tecnologías utilizadas en el proyecto

Tabla 41: Lista de Herramientas y tecnologías utilizados en el proyecto.

Herramienta/ Tecnología	Ubicación	Propósito
Word	https://products.office.com/es-es/word	Procesador de texto utilizado en la redacción del Trabajo Fin de Master.
Draw.io	https://www.draw.io/	Aplicación de diagramación Online utilizado en la creación de los diagramas del sistema.
Syntax Highlight Code	http://www.planetb.ca/syntax-highlight-word	Utilizado para el resaltado de código fuente descrito en la memoria del Master.
Zotero	https://www.zotero.org	Gestor de referencias bibliográficas usado para la creación de la Bibliografía.
TortoiseSVN	https://tortoisessvn.net/downloads.html	Es un cliente de subversión y se utilizó para versionar archivos generados durante la elaboración del proyecto (bibliografías, código fuente, documentos, tutoriales)
Git	https://git-scm.com/download/win	Es un sistema de control de versiones distribuidos y se utilizó como repositorio de código fuente local de las aplicaciones desarrolladas.
GitHub	https://github.com	Es un servicio de alojamiento de repositorio Git y se utilizó como repositorio remoto del código fuente elaborado.
Redmine	http://core.ugr.es/redmine/	Es una herramienta para la gestión de proyectos

		del grupo de investigación en Sistemas Concurrentes, fue utilizado para la gestión y seguimiento del todo el proyectos implementado durante el Master.
Android Studio 2.3.3	https://developer.android.com/studio/index.html?hl=es-419	Es un entorno de desarrollo integrado oficial para la plataforma Android. Este IDE fue utilizado para el desarrollo de la aplicación Android Things y la aplicación móvil del usuario final.
Raspberry PI 3	https://www.raspberrypi.org/products/raspberry-pi-3-model-b/	Es un computador de placa reducida y se utilizó como base para la implementación del Sistema Empotrado quien es el encargado de obtener la temperatura y presión atmosférica.
Rainbow HAT	https://shop.pimoroni.com/products/rainbow-hat-for-android-things	Es una placa de circuito que incluye varios sensores; se utilizó para la obtención de la temperatura y presión atmosférica.
SDFormatter V4.0	https://sd-card-formatter.uptodown.com/windows	Es un software que proporciona un acceso sencillo a todos los formatos de la tarjeta de memoria SD. Fue utilizado para preparar la tarjeta SD.
Win32 Disk Imagen Version 0.9.5	https://sourceforge.net/projects/win32diskimager/	Es una sencilla aplicación que permite grabar imágenes en memorias USB o tarjetas SD. Se usó para grabar la imagen de Android Things en la tarjeta SD del Raspberry.
Protoboard	https://es.wikipedia.org/wiki/Placa_de_pruebas	Es un tablero para armar circuitos electrónicos y se utilizó en el armado de prototipos.
Eclipse Neon 2 Release 4.6.2	http://www.eclipse.org/downloads/packages/eclipse-ide-jee-developers/neon2	Es un entorno de desarrollo, el mismo fue utilizado para la implementación del programa Cloud DataFlow.
Java	http://www.oracle.com/technet/work/java/javase/downloads/jdk8-downloads-2133151.html	El lenguaje de programación java fue usado para la implementación de todas las aplicaciones del proyecto.
Android	https://developer.android.com/sdk/older_releases.html	El sistema operativo Android se utilizó para la ejecución de la aplicación móvil del proyecto.
Material Design	https://developer.android.com/design/material/index.html?hl=es-419	Es una guía integral para el diseño visual y fue manejado en el diseño de la aplicación móvil del usuario final.
Maven	https://mvnrepository.com/	Es una herramienta de software para gestión y construcción de proyectos Java; en el proyecto se manejó para la gestión de dependencias de la aplicación Cloud Dataflow.
Gradle	https://docs.gradle.org/current/release-notes.html	Es una herramienta de compilación creado por Google para la gestión y construcción de proyectos Android; en el proyecto se usó para la gestión de dependencias de la aplicación móvil.
IoT Developer Console Weave	https://nest.com/weave/	Es una aplicación web que proporciona herramientas para acceder a dispositivos, realizar

		acciones de configuración y monitorización. En su momento se utilizó para la configuración de dispositivos Weave.
Android Things	https://partner.android.com/things/console/?pli=1	Es un sistema operativo del IoT fue utilizado para la gestión y ejecución de la aplicación encargada de obtener la temperatura y presión atmosférica medioambiental.
Google Cloud Platform	https://cloud.google.com/	La plataforma de computación en la nube de Google se usó para el procesamiento de los datos recolectados por los Sistemas Empotrados.
Cloud Pub/Sub	https://console.cloud.google.com/cloudpubsub/	El servicio de mensajería de Google fue utilizado para el almacenamiento temporal de datos recolectados por los sensores.
Cloud Dataflow	https://console.cloud.google.com/dataflow	El servicio de procesamiento de datos de Google se manejó para la transferencias de los mensajes del servicio Cloud Pub/Sub hacia el servicio Cloud BigQuery.
Cloud BigQuery	https://bigquery.cloud.google.com/	El servicio data warehouse fue utilizado para el almacenamiento de la información generada en el sistema medioambiental.
Cloud Compute Engine	https://console.cloud.google.com/compute/	El servicio de administración y ejecución de maquina virtuales fue utilizado para alojar el servidor Debian.
Cloud Shell	https://cloud.google.com/shell/	La interfaz web de Google Cloud Platform se manejó para la ejecución de línea de comandos.
Cloud Storage	https://console.cloud.google.com/storage/	El servicio de almacenamiento de datos se utilizó en la implementación de pruebas de integración para el acumulación de logs de ejecución.
Apache Beam	https://beam.apache.org/	La biblioteca para el procesamiento de datos masivos fue usado en la manipulación de los mensajes entre los servicios Cloud Pub/Sub y BigQuery.
Firebase	https://firebase.google.com/	La plataforma en la nube Firebase se manejó para el envío de notificaciones push hacia las aplicaciones móviles.
Postman	https://www.getpostman.com/	El cliente HTTP se utilizó para gestionar el servicio web de Firebase.
Retrofit	http://square.github.io/retrofit/	El cliente REST se utilizó en la implementación del servicio web para enviar peticiones hacia Firebase.
Jackson	https://github.com/FasterXML/jackson-core	Librería utilizada en la aplicación de Cloud Dataflow se empleó deserializar los mensajes del servicio Cloud Pub/Sub.

Anexo D - Sistemas Operativos IoT de Código Abierto

Tabla 42: Tabla comparativa de Sistemas Operativos IoT de Código Abierto.

Característica	Contiki	RIOT	FreeRTOS	TinyOS	uClinux	Mbed	Android Things
Arquitectura	Monolítico	Microkernel RTOS	Microkernel RTOS	Monolítico	Monolítico	Monolítico	Monolítico
Modelo de Programación	Dirigido por eventos	multihilo	multihilo	Dirigido por eventos	Multihilo	Dirigido por eventos e hilo simple	Multihilo
Clase de dispositivos	Clase 0, 1	Clase 1, 2	Clase 1, 2	Clase 0	>Clase 2	Clase 1, 2	>Clase 2
Empresas o proveedores de microcontroladores compatibles	AVR, MSP430, ARM, Cortex-M, PIC32, x86	AVR, MSP430, ARM, Cortex-M, px27ax	AVR, MP430, ARM, x86, 8052, Renesas	AVR, MSP430, px27ax	ARM7, ARM, Cortex-M	ARM, Cortex-A7, ARM Cortex A53, Intel Atom	ARM Cortex A7, ARM Cortex A53, Intel Atom
Lenguajes de Programación	C	C, C++	C	nesC	C	C, C++	Java, C, C++
Licencia	BSD	GPLv2	GPL	BSD	GPLv2	Apache 2.0	Apache 2.0 y GNU GPL